Palo Alto Research Centers

The Ethernet Local Network: Three Reports

XEROX

The Ethernet Local Network: Three Reports

CSL-80-2 February 1980, reprinted October 1982 [P80-00013]

© Copyright Xerox Corporation 1982. All rights reserved.

Abstract: This report reproduces three previously published papers on the Ethernet local-area communications network:

Ethernet: Distributed Packet Switching for Local Computer Networks, by Robert M. Metcalfe and David R. Boggs. Appeared in *Communications of the ACM*, vol. 19 no. 7, July 1976.

Practical Considerations in Ethernet Local Network Design, by Ronald C. Crane and Edward A. Taft. Presented at the *Hawaii International Conference on System Sciences*, January 1980.

Measured Performance of an Ethernet Local Network, by John F. Shoch and Jon A. Hupp. A preliminary version was presented at the *Local Area Communications Network Symposium*, Boston, May 1979.

The first paper introduces the Ethernet-style network, a multi-access broadcast packet-switched communication system, and presents the theory (CSMA/CD) that underlies it. The second paper discusses the design and implementation of the prototype Ethernet system, and the final paper presents performance results based on several years' practical experience with that system.

XEROX

Xerox Corporation
Palo Alto Research Centers
3333 Coyote Hill Road
Palo Alto, California 94304

Ethernet: Distributed Packet Switching for Local Computer Networks

by Robert M. Metcalfe and David R. Boggs

CSL-75-7 May 1975, reprinted October 1982 [P76-00002]

© Copyright Association for Computing Machinery 1976. Printed with permission.

Abstract: Ethernet is a branching broadcast communication system for carrying digital data packets among locally distributed computing stations. The packet transport mechanism provided by Ethernet has been used to build systems which can be viewed as either local computer networks or loosely coupled multiprocessors.

An Ethernet's shared communication facility, its Ether, is a passive broadcast medium with no central control. Coordination of access to the Ether for packet broadcasts is distributed among the contending transmitting stations using controlled statistical arbitration. Switching of packets to their destinations on the Ether is distributed among the receiving stations using packet address recognition.

Design principles and implementation are described based on experience with an operating Ethernet of 100 nodes along a kilometer of coaxial cable. A model for estimating performance under heavy loads and a packet protocol for error-controlled communication are included for completeness.

A version of this paper appeared in *Communications of the ACM*, vol. 19 no. 7, July 1976.

CR categories: 3.81, 4.32, 6.35.

Key words and phrases: computer networks, packet switching, multiprocessing, distributed control, distributed computing, broadcast communication, statistical arbitration.

XEROX

Xerox Corporation
Palo Alto Research Centers
3333 Coyote Hill Road
Palo Alto, California 94304

1. Background

One can characterize distributed computing as a spectrum of activities varying in the degree of decentralization, with one extreme being remote computer networking and the other extreme being multiprocessing. Remote computer networking is the loose interconnection of previously isolated, widely separated, and rather large computing systems. Multiprocessing is the construction of previously monolithic and serial computing systems from increasingly numerous and smaller pieces computing in parallel. Near the middle of this spectrum is local networking, the interconnection of computers to gain the resource sharing of computer networking and the parallelism of multiprocessing.

The separation between computers and the associated bit rate of their communication can be used to divide the distributed computing spectrum into broad activities. The product of separation and bit rate, now about 1 gigabit-meter per second (1 Gbmps), is an indication of the limit of current communication technology and can be expected to increase with time.

Activity	Separation	Bit Rate
Remote Networks	> 10 km	< .1 Mbps
Local Networks	10-.1 km	.1-10 Mbps
Multiprocessors	< .1 km	> 10 Mbps

1.1 Remote Computer Networking

Computer networking evolved from telecommunications, *terminal-computer* communication, where the object was to connect remote terminals to a central computing facility. As the need for *computer-computer* interconnection grew, computers themselves were used to provide communication [Baran, 1964; Rustin, 1972; Abramson, 1975]. Communication *using* computers as packet switches [Roberts, 1970; Heart, 1970, 1973; Metcalfe 1973b] and communications *among* computers for resource sharing [Crocker, 1972; Thomas, 1973] were both advanced by the development of the Arpa Computer Network.

The Aloha Network at the University of Hawaii was originally developed to apply packet radio techniques for communication between a central computer and its terminals scattered among the Hawaiian Islands [Abramson, 1970, 1975]. Many of the terminals are now minicomputers communicating among themselves using the Aloha Network's Menehune as a packet switch. The Menehune and an Arpanet Imp are now connected providing terminals on the Aloha Network access to computing resources on the U.S. mainland.

Just as computer networks have grown across continents and oceans to interconnect major computing facilities around the world, they are now growing down corridors and between buildings to interconnect minicomputers in offices and laboratories [Ashenhurst, 1975; Willard, 1973; Fraser, 1975; Farber, 1973, 1975].

1.2 Multiprocessing

Multiprocessing first took the form of connecting an I/O controller to a large central computer; IBM's ASP is a classic example [Rustin, 1972]. Next, multiple central processors were connected to a common memory to provide more power for compute-bound applications [Thornton, 1970]. For certain of these applications, more exotic multiprocessor architectures such as Illiac IV were introduced [Barnes, 1968].

More recently minicomputers have been connected in multiprocessor configurations for economy, reliability, and increased system modularity [Wulf, 1972; Ornstein, 1975]. The trend has been toward decentralization for reliability; loosely coupled multiprocessor systems depend less on shared central memory and more on *thin wires* for interprocess communication with increased component isolation [Metcalfe, 1972a, 1973b]. With the continued thinning of interprocessor communication for reliability and the development of distributable applications, multiprocessing is gradually approaching a local form of distributed computing.

1.3 Local Computer Networking

Ethernet shares many objectives with other local networks such as Mitre's Mitrix, Bell Telephone Laboratory's Spider, and U.C. Irvine's Distributed Computing System (DCS) [Willard, 1973; Fraser, 1975; Farber, 1973, 1975]. Prototypes of all four local networking schemes operate at bit rates between one and three megabits per second. Mitrix and Spider have a central minicomputer for switching and bandwidth allocation while DCS and Ethernet use distributed control. Spider and DCS use a ring communication path, Mitrix uses off-the-shelf CATV technology to implement two one-way busses, and our experimental Ethernet uses a branching two-way passive bus. Differences among these systems are due to differences among their intended applications, differences among the cost constraints under which trade-offs were made, and differences of opinion among researchers.

Before going into a detailed description of Ethernet, we offer the following overview (see Figure 1).

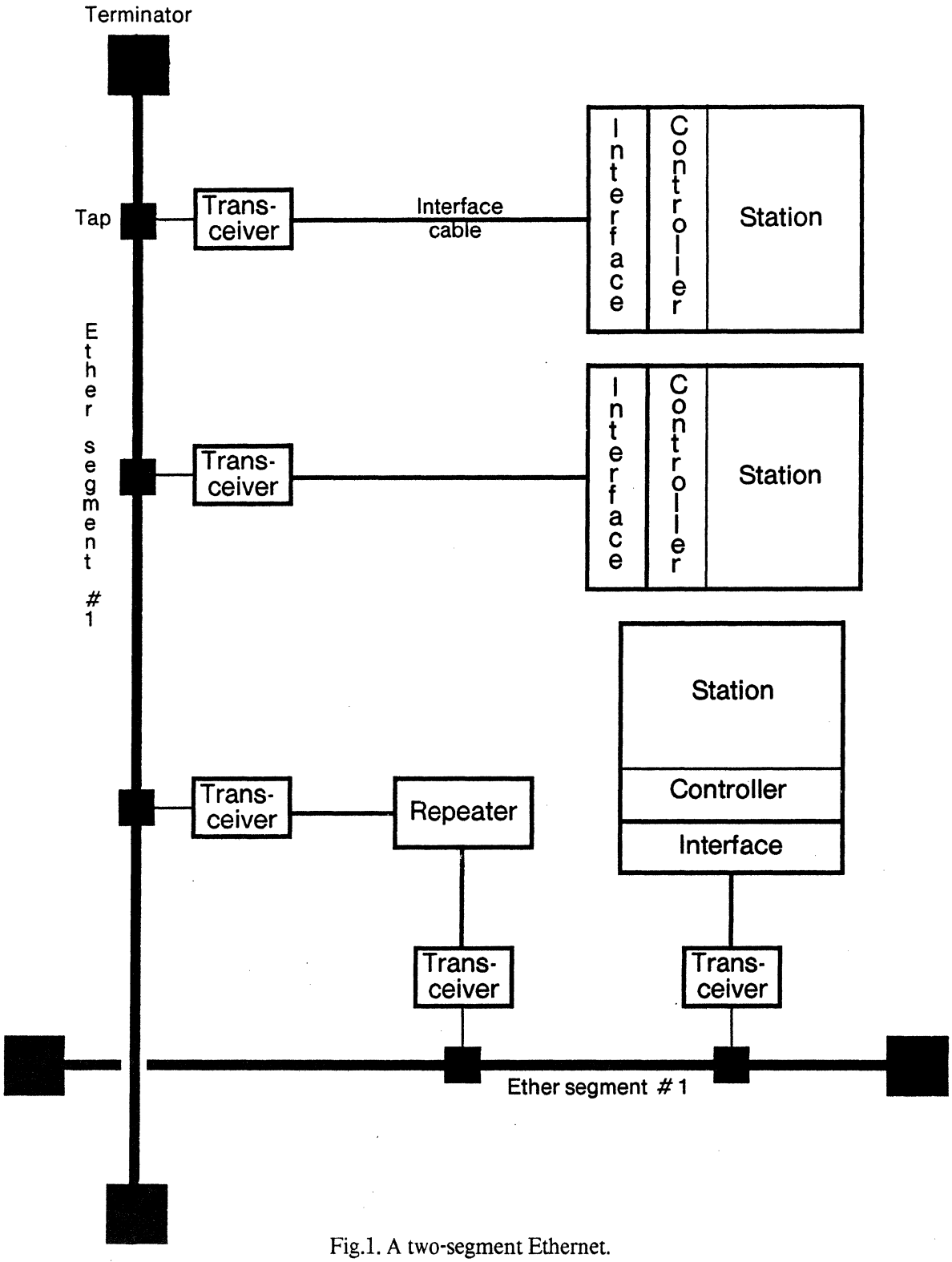


Fig.1. A two-segment Ethernet.

2. System Summary

Ethernet is a system for local communication among computing stations. Our experimental Ethernet uses tapped coaxial cables to carry variable-length digital data packets among, for example, personal minicomputers, printing facilities, large file storage devices, magnetic tape backup stations, larger central computers, and longer-haul communication equipment.

The shared communication facility, a branching Ether, is passive. A station's Ethernet interface connects bit-serially through an interface cable to a transceiver which in turn taps into the passing Ether. A packet is broadcast onto the Ether, is heard by all stations, and is copied from the Ether by destinations which select it according to the packet's leading address bits. This is broadcast packet switching and should be distinguished from store-and-forward packet switching in which routing is performed by intermediate processing elements. To handle the demands of growth, an Ethernet can be extended using packet repeaters for signal regeneration, packet filters for traffic localization, and packet gateways for internetwork address extension.

Control is completely distributed among stations with packet transmissions coordinated through statistical arbitration. Transmissions initiated by a station defer to any which may already be in progress. Once started, if interference with other packets is detected, a transmission is aborted and rescheduled by its source station. After a certain period of interference-free transmission, a packet is heard by all stations and will run to completion without interference. Ethernet controllers in colliding stations each generate random retransmission intervals to avoid repeated collisions. The mean of a packet's retransmission intervals is adjusted as a function of collision history to keep Ether utilization near the optimum with changing network load.

Even when transmitted without source-detected interference, a packet may still not reach its destination without error; thus, packets are delivered *only with high probability.* Stations requiring a residual error rate lower than that provided by the bare Ethernet packet transport mechanism must follow mutually agreed upon packet protocols.

3. Design Principles

Our object is to design a communication system which can grow smoothly to accommodate several buildings full of personal computers and the facilities needed for their support.

Like the computing stations to be connected, the communication system must be inexpensive. We choose to distribute control of the communications facility among the communicating computers to eliminate the reliability problems of an active central controller, to avoid creating a bottleneck in a system rich in parallelism, and to reduce the fixed costs which make small systems uneconomical.

Ethernet design started with the basic idea of packet collision and retransmission developed in the Aloha Network [Abramson, 1970]. We expected that, like the Aloha Network, Ethernets would carry bursty traffic so that conventional synchronous time-division multiplexing (STDM) would be inefficient [Roberts, 1970; Abramson, 1970, 1975; Metcalfe, 1973b]. We saw promise in the Aloha

approach to distributed control of radio channel multiplexing and hoped that it could be applied effectively with media suited to local computer communication. With several innovations of our own, the promise is realized.

Ethernet is named for the historical *luminiferous ether* through which electromagnetic radiations were once alleged to propagate. Like an Aloha radio transmitter, an Ethernet transmitter broadcasts completely-addressed transmitter-synchronous bit sequences called packets onto the Ether and hopes that they are heard by the intended receivers. The Ether is a logically passive medium for the propagation of digital signals and can be constructed using any number of media including coaxial cables, twisted pairs, and optical fibers.

3.1 Topology

We cannot afford the redundant connections and dynamic routing of store-and-forward packet switching to assure reliable communication, so we choose to achieve reliability through simplicity. We choose to make the shared communication facility passive so that the failure of an active element will tend to affect the communications of only a single station. The layout and changing needs of office and laboratory buildings leads us to pick a network topology with the potential for convenient incremental extention and reconfiguration with minimal service disruption.

The topology of the Ethernet is that of an unrooted tree. It is a *tree* so that the Ether can branch at the entrance to a building's corridor, yet avoid multipath interference. There must be only one path through the Ether between any source and destination; if more than one path were to exist, a transmission would interfere with itself, repeatedly arriving at its intended destination having travelled by paths of different length. The Ether is *unrooted* because it can be extended from any of its points in any direction. Any station wishing to join an Ethernet taps into the Ether at the nearest convenient point.

Looking at the relationship of interconnection and control, we see that Ethernet is the dual of a star network. Rather than *distributed* interconnection through many separate links and *central* control in a switching node, as in a star network, the Ethernet has *central* interconnection through the Ether and *distributed* control among its stations.

Unlike an Aloha Network which is a star network with an outgoing broadcast channel and an incoming multi-access channel, an Ethernet supports many-to-many communication with a single broadcast multi-access channel.

3.2 Control

Sharing of the Ether is controlled in such a way that it is not only possible but probable that two or more stations will attempt to transmit a packet at roughly the same time. Packets which overlap in time on the Ether are said to *collide*; they interfere so as to be unrecognizable by a receiver. A station recovers from a detected collision by abandoning the attempt and retransmitting

the packet after some dynamically-chosen random time period. Arbitration of conflicting transmission demands is both distributed and statistical.

When the Ether is largely unused, a station transmits its packets at will, the packets are received without error, and all is well. As more stations begin to transmit, the rate of packet interference increases. Ethernet controllers in each station are built to adjust the mean retransmission interval in proportion to the frequency of collisions; sharing of the Ether among competing station-station transmissions is thereby kept near the optimum [Metcalfe, 1973a, 1973b].

A degree of cooperation among the stations is required to share the Ether equitably. In demanding applications certain stations might usefully take transmission priority through some systematic violation of equity rules. A station could usurp the Ether by not adjusting its retransmission interval with increasing traffic or by sending very large packets. Both practices are now prohibited by low-level software in each station.

3.3 Addressing

Each packet has a source and destination, both of which are identified in the packet's header. A packet placed on the Ether eventually propagates to all stations. Any station can copy a packet from the Ether into its local memory, but normally only an active destination station matching its address in the packet's header will do so as the packet passes. By convention, a zero destination address is a wildcard and matches all addresses; a packet with a destination of zero is called a *broadcast packet.*

3.4 Reliability

An Ethernet is probabilistic. Packets may be lost due to interference with other packets, impulse noise on the Ether, an inactive receiver at a packet's intended destination, or purposeful discard. Protocols used to communicate through an Ethernet must assume that packets will be received correctly at intended destinations *only with high probability.*

An Ethernet gives its *best efforts* to transmit packets successfully, but it is the responsibility of processes in the source and destination stations to take the precautions necessary to assure reliable communication of the quality they themselves desire [Metcalfe, 1972a, 1973b]. Recognizing the costliness and dangers of promising "error-free" communication, we refrain from guaranteeing reliable delivery of any single packet to get both economy of transmission and high reliability averaged over many packets [Metcalfe, 1973b]. Removing the responsibility for reliable communication from the packet transport mechanism allows us to tailor reliability to the application and to place error recovery where it will do the most good. This policy becomes more important as Ethernets are interconnected in a hierarchy of networks through which packets must travel farther and suffer greater risks.

3.5 Mechanisms

A station connects to the Ether with a *tap* and a *transceiver.* A tap is a device for phyically connecting to the Ether while disturbing its transmission characteristics as little as possible. The design of the transceiver must be an exercise in paranoia. Precautions must be taken to insure that likely failures in the transceiver or station do not result in pollution of the Ether. In particular, removing power from the transceiver should cause it to disconnect from the Ether.

Five mechanisms are provided in our experimental Ethernet for reducing the probability and cost of losing a packet. These are (1) carrier detection, (2) interference detection, (3) packet error detection, (4) truncated packet filtering, and (5) collision consensus enforcement.

3.5.1 Carrier Detection. As a packet's bits are placed on the Ether by a station, they are phase encoded, (like bits on a magnetic tape) which guarantees that there is at least one transition on the Ether during each bit time. The passing of a packet on the Ether can therefore be detected by listening for its transitions. To use a radio analogy, we speak of the presence of *carrier* as a packet passes a transceiver. Because a station can sense the carrier of a passing packet, it can delay sending one of its own until the detected packet passes safely. The Aloha Network does not have carrier detection and consequently suffers a substantially higher collision rate. Without carrier detection, efficient use of the Ether would decrease with increasing packet length. In Section 6 below, we show that with carrier detection, Ether efficiency increases with increasing packet length.

With carrier detection we are able to implement *deference:* no station will start transmitting while hearing carrier. With deference comes *acquisition:* once a packet transmission has been in progress for an Ether end-to-end propagation time, all stations are hearing carrier and are deferring; the Ether has been acquired and the transmission will complete without an interfering collision.

With carrier detection, collisions should occur only when two or more stations find the Ether silent and begin transmitting simultaneously: within an Ether end-to-end propagation time. This will almost always happen immediately after a packet transmission during which two or more stations were deferring. Because stations do not now randomize after deferring, when the transmission terminates, the waiting stations pile on together, collide, randomize, and retransmit.

3.5.2 Interference Detection. Each transceiver has an interference detector. Interference is indicated when the transceiver notices a difference between the value of the bit it is receiving from the Ether and the value of the bit it is attempting to transmit.

Interference detection has three advantages. First, a station detecting a collision knows that its packet has been damaged. The packet can be scheduled for retransmission immediately, avoiding a long acknowledgement timeout. Second, interference periods on the Ether are limited to a maximum of one round trip time. Colliding packets in the the Aloha Network run to completion, but the truncated packets resulting from Ethernet collisions waste only a small fraction of a packet time on the Ether. Third, the frequency of detected interference is used to estimate Ether traffic for adjusting retransmission intervals and optimizing channel efficiency.

3.5.3 Packet Error Detection. As a packet is placed on the Ether, a checksum is computed and appended. As the packet is read from the Ether, the checksum is recomputed. Packets which do not carry a consistent checksum are discarded. In this way transmission errors, impulse noise errors, and errors due to undetected interference are caught at a packet's destination.

3.5.4 Truncated Packet Filtering. Interference detection and deference cause most collisions to result in *truncated packets* of only a few bits; colliding stations detect interference and abort transmission within an Ether round-trip time. To reduce the processing load that the rejection of such obviously damaged packets would place on listening station software, truncated packets are filtered out in hardware.

3.5.5 Collision Consensus Enforcement. When a station determines that its transmission is experiencing interference, it momentarily jams the Ether to insure that all other participants in the collision will detect interference and, because of deference, will be forced to abort. Without this *collision consensus enforcement* mechanism, it is possible that the transmitting station which would otherwise be the last to detect a collision might not do so as the other interfering transmissions successively abort and stop interfering. Although the packet may look good to that last transmitter, different path lengths between the colliding transmitters and the intended receiver will cause the packet to arrive damaged.

4. Implementation

Our choices of 1 kilometer, 3 megabits per second, and 256 stations for the parameters of an experimental Ethernet were based on characteristics of the locally-distributed computer communication environment and our assessments of what would be marginally achievable; they were certainly not hard restrictions essential to the Ethernet concept.

We expected that a reasonable maximum network size would be on the order of 1 kilometer of cable. We used this working number to choose among Ethers of varying signal attenuation and to design transceivers with appropriate power and sensitivity.

The dominant station on our experimental Ethernet is a minicomputer for which 3 megabits per second is a convenient data transfer rate. By keeping the peak rate well below that of the computer's path to main memory, we reduce the need for expensive special-purpose packet buffering in our Ethernet interfaces. By keeping the peak rate as high as is convenient, we provide for larger numbers of stations and more ambitious multiprocessing communications applications.

To expedite low-level packet handling among 256 stations, we allocate the first 8-bit byte of the packet to be the destination address field and the second byte to be the source address field (see Figure 2). 256 is a number small enough to allow each station to get an adequate share of the available bandwidth and approaches the limit of what we can achieve with current techniques for tapping cables. 256 is only a convenient number for the lowest level of protocol; higher levels can

accomodate extended address spaces with additional fields inside the packet and software to interpret them.

Our experimental Ethernet implementation has four major parts: the Ether, transceivers, interfaces, and controllers (see Figure 1).

4.1 Ether

We chose to implement our experimental Ether using low-loss coaxial cable with off-the-shelf CATV taps and connectors. It is possible to mix Ethers on a single Ethernet; we use a smaller-diameter coax for convenient connection within station clusters and a larger-diameter coax for low-loss runs between clusters. The cost of coaxial cable Ether is insignificant relative to the cost of the distributed computing systems supported by Ethernet.

4.2 Transceivers

Our experimental transceivers can drive a kilometer of coaxial cable Ether tapped by 256 stations transmitting at 3 megabits per second. The transceivers can *endure* (i.e., work after) sustained direct shorting, improper termination of the Ether, and simultaneous drive by all 256 stations; they can *tolerate* (i.e., work during) ground differentials and everyday electrical noise, from typewriters or electric drills, encountered when stations are separated by as much as a kilometer.

An Ethernet transceiver attaches directly to the Ether which passes by in the ceiling or under the floor. It is powered and controlled through 5 twisted pairs in an interface cable carrying transmit data, receive data, interference detect, and power supply voltages. When unpowered, the transceiver disconnects itself electrically from the Ether. Here is where our fight for reliability is won or lost; a broken transceiver can, but should not, bring down an entire Ethernet. A watchdog timer circuit in each transceiver attempts to prevent pollution of the Ether by shutting down the output stage if it acts suspiciously. For transceiver simplicity we use the Ether's base frequency band, but an Ethernet could be built to use any suitably sized band of a frequency division multiplexed Ether.

Even though our experimental transceivers are very simple and can tolerate only limited signal attenuation, they have proven quite adequate and reliable. A more sophisticated transceiver design might permit passive branching of the Ether and wider station separation.

4.3 Interface

An Ethernet interface serializes and deserializes the parallel data used by its station. There are a number of different stations on our Ethernet; an interface must be built for each kind.

Each interface is equipped with the hardware necessary to compute a 16-bit cyclic redundancy checksum (CRC) on serial data as it is transmitted and received. This checksum only protects against errors in the Ether and specifically not errors in the parallel portions of the interface hardware or station. Higher-level software checksums are recommended for applications in which a higher degree of reliability is required.

A transmitting interface uses a packet buffer address and word count to serialize and phase encode a variable number of 16-bit words which are taken from the station's memory and passed to the transceiver, preceded by a start bit (called SYNC in Figure 2) and followed by the CRC. A receiving interface uses the appearance of carrier to detect the start of a packet and uses the SYNC bit to aquire bit phase. As long as carrier stays on, the interface decodes and deserializes the incoming bit stream depositing 16-bit words in a packet buffer in the station's main memory. When carrier goes away, the interface checks that an integral number of 16-bit words has been received and that the CRC is correct. The last word received is assumed to be the CRC and is not copied into the packet buffer.

These interfaces ordinarily include hardware for accepting only those packets with appropriate addresses in their headers. Hardware address filtering helps a station avoid burdensome software packet processing when the Ether is very busy carrying traffic intended for other stations.

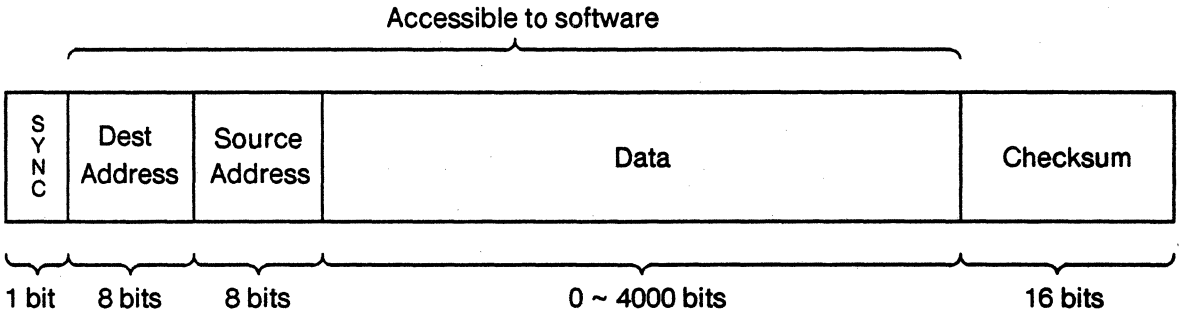


Fig. 2. Ethernet packet layout.

4.4 Controller

An Ethernet controller is the station-specific low-level firmware or software for getting packets onto and out of the Ether. When a source-detected collision occurs, it is the source controller's responsibility to generate a new random retransmission interval based on the updated collision count. We have studied a number of algorithms for controlling retransmission rates in stations to maintain Ether efficiency [Metcalfe, 1973a, 1974]. The more practical of these algorithms estimate traffic load using recent collision history.

Retransmission intervals are multiples of a *slot*, the maximum time between starting a transmission and detecting a collision, one end-to-end round trip delay. An Ethernet controller begins transmission of each new packet with a mean retransmission interval of one slot. Each time a transmission attempt ends in collision, the controller delays for an interval of random length with a mean twice that of the previous interval, defers to any passing packet, and then attempts retransmission. This heuristic approximates an algorithm we have called Binary Exponential Backoff (see Figure 3) [Metcalfe, 1974].

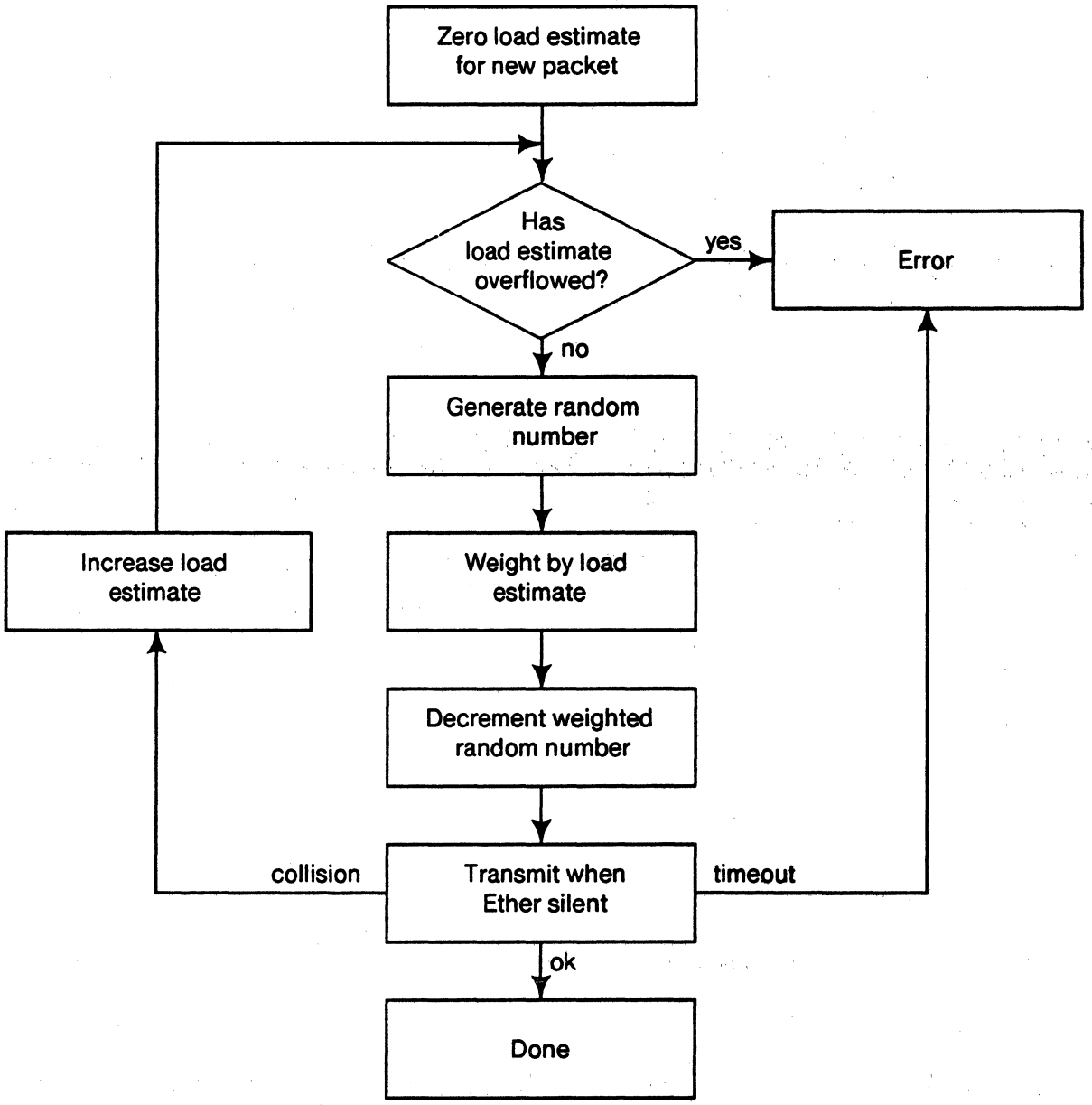


Fig. 3. Collision control algorithm.

When the network is unloaded and collisions are rare, the mean seldom departs from one and retransmissions are prompt. As the traffic load increases, more collisions are experienced, a backlog of packets builds up in the stations, retransmission intervals increase, and retransmission traffic backs off to sustain channel efficiency.

5. Growth

5.1 Signal Cover

One can expand an Ethernet just so far by adding transceivers and Ether. At some point, the transceivers and Ether will be unable to carry the required signals. The *signal cover* can be extended with a simple unbuffered *packet repeater.* In our experimental Ethernet where, because of transceiver simplicity the Ether cannot be branched passively, a simple repeater may join any number of Ether *segments* to enrich the topology while extending the signal cover.

We operate an experimental two-segment packet repeater, but hope to avoid relying on them. In branching the Ether and extending its signal cover, there is a trade-off between using sophisticated transceivers and using repeaters. With increased power and sensitivity, transceivers become more expensive and less reliable. The introduction of repeaters into an Ethernet makes the centrally interconnecting Ether active. The failure of a transceiver will sever the communications of its owner; the failure of a repeater partitions the Ether severing many communications.

5.2 Traffic Cover

One can expand an Ethernet just so far by adding Ether and packet repeaters. At some point the Ether will be so busy that additional stations will just divide more finely the already inadequate bandwidth. The *traffic cover* can be extended with an unbuffered traffic-filtering repeater or *packet filter,* which passes packets from one Ether segment to another only if the destination station is located on the new segment. A packet filter also extends the signal cover.

5.3 Address Cover

One can expand an Ethernet just so far by adding Ether, repeaters, and traffic filters. At some point there will be too many stations to be addressed with the Ethernet's 8-bit addresses. The *address cover* can be extended with *packet gateways* and the software addressing conventions they implement [Cerf, 1974]. Addresses can be expanded in two directions: *down* into the station by adding fields to identify destination ports or processes within a station, and *up* into the internetwork by adding fields to identify destination stations on remote networks. A gateway also extends the traffic and signal covers.

There can be only one repeater or packet filter connecting two Ether segments; a packet repeated onto a segment by multiple repeaters would interfere with itself. However, there is no limit to the number of gateways connecting two segments; a gateway only repeats packets addressed to itself as an intermediary. Failure of the single repeater connecting two segments partitions the network; failure of a gateway need not partition the net if there are paths through other gateways between the segments.

6. Performance

Here is a simple set of formulas with which to characterize the performance expected of an Ethernet when it is heavily loaded. More elaborate analyses and several detailed simulations have been done, but the following simple model has proven very useful in understanding the Ethernet's distributed contention scheme, even when it is loaded beyond expectations [Abramson, 1970; Metcalfe, 1973a, 1973b, 1974; Roberts, 1973; Murthy, 1975].

We develop a simple model of the performance of a loaded Ethernet by examining alternating Ether time periods. The first, called a *transmission interval,* is that during which the Ether has been acquired for a successful packet transmission. The second, called a *contention interval,* is that composed of the retransmission slots of Section 4.4, during which stations attempt to acquire control of the Ether. Because the model's Ethernets are loaded and because stations defer to passing packets before starting transmission, the slots are synchronized by the tail of the preceding acquisition interval. A slot will be empty when no station chooses to attempt transmission in it and it will contain a collision if more than one station attempts to transmit. When a slot contains only one attempted transmission, then the Ether has been acquired for the duration of a packet, the contention interval ends, and a transmission interval begins.

Let P be the number of bits in an Ethernet *packet.* Let C be the peak *capacity* in bits per second, carried on the Ether. Let T be the *time* in seconds of a slot, the number of seconds it takes to detect a collision after starting a transmission. Let us assume that there are Q stations continuously *queued* to transmit a packet; either the acquiring station has a new packet immediately after a successful acquisition or another station comes ready. Note that Q also happens to give the total *offered load* on the network which for this analysis is always 1 or greater. We assume that a queued station attempts to transmit in the current slot with probability $1/Q$, or delays with probability $1-(1/Q)$; this is known to be the optimum statistical decision rule, approximated in Ethernet stations by means of our load-estimating retransmission control algorithms [Metcalfe, 1973a, 1973b].

6.1 Acquisition Probability

We now compute A, the probability that exactly one station attempts a transmission in a slot and therefore *acquires* the Ether. A is $Q*(1/Q)*((1-(1/Q))^{(Q-1)})$; there are Q ways in which one station can choose to transmit (with probability $(1/Q)$) while Q-1 stations choose to wait (with probability $1-(1/Q)$). Simplifying,

$$A = (1-(1/Q))^{(Q-1)}.$$

6.2 Waiting Time

We now compute W, the mean number of slots of *waiting* in a contention interval before a successful acquisition of the Ether by a station's transmission. The probability of waiting no time at all is just A, the probability that one and only one station chooses to transmit in the first slot following a transmission. The probability of waiting 1 slot is $A*(1-A)$; the probability of waiting i slots is $A*((1-A)^i)$. The mean of this geometric distribution is

$$W = (1-A)/A.$$

6.3 Efficiency

We now compute E, that fraction of time the Ether is carrying good packets, the *efficiency*. The Ether's time is divided between transmission intervals and contention intervals. A packet transmission takes P/C seconds. The mean time to acquisition is $W*T$. Therefore, by our simple model,

$$E = (P/C)/((P/C)+(W*T)).$$

Table I presents representative performance figures (i.e., E) for our experimental Ethernet with the indicated packet sizes and number of *continuously queued* stations. The efficiency figures given do not account for inevitable reductions due to headers and control packets nor for losses due to imprecise control of the retransmission parameter $1/Q$; the former is straightforwardly protocol-dependent and the latter requires analysis beyond the scope of this paper. Again, we feel that all of the Ethernets in the table are overloaded; normally loaded Ethernets will usually have a Q much less than 1 and exhibit behavior not covered by this model.

For our calculations, we use a C of 3 megabits per second and a T of 16 microseconds. The slot duration, T, must be long enough to allow a collision to be detected or at least twice the Ether's round trip time. We limit in software the maximum length of our packets to be near 4000 bits to keep the latency of network access down and to permit efficient use of station packet buffer storage.

Table I. Ethernet Efficiency.

Q	$P=4096$	$P=1024$	$P=512$	$P=48$
1	1.0000	1.0000	1.0000	1.0000
2	0.9884	0.9552	0.9143	0.5000
3	0.9857	0.9447	0.8951	0.4444
4	0.9842	0.9396	0.8862	0.4219
5	0.9834	0.9367	0.8810	0.4096
10	0.9818	0.9310	0.8709	0.3874
32	0.9807	0.9272	0.8642	0.3737
64	0.9805	0.9263	0.8627	0.3708
128	0.9804	0.9259	0.8620	0.3693
256	0.9803	0.9257	0.8616	0.3686

For packets whose size is above 4000 bits, the efficiency of our experimental Ethernet stays well above 95 percent. For packets with a size approximating that of a slot, Ethernet efficiency approaches 1/e, the asymptotic efficiency of a slotted Aloha network [Roberts, 1973].

7. Protocol

There is more to the construction of a viable packet communication system than simply providing the mechanisms for packet transport. Methods for error correction, flow control, process naming, security, and accounting must also be provided through higher level protocols implemented on top of the Ether control protocol decribed in Sections 3 and 4 above [Cerf, 1974; Crocker, 1972; Metcalfe, 1973b; Farber, 1975; Rowe, 1975; Walden, 1972]. Ether control includes packet framing, error detection, addressing and multi-access control; like other line control procedures, Ethernet is used to support numerous network and multiprocessor architectures [SDLC, 1974; SNA, 1975].

Here is a brief description of one simple error-controlling packet protocol. The EFTP (Ethernet File Transfer Protocol) is of interest both because it is relatively easy to understand and implement correctly and because it has dutifully carried many valuable files during the development of more general and efficient protocols.

7.1 General Terminology

In discussing packet protocols, we use the following generally useful terminology. A packet is said to have a *source* and a *destination.* A flow of data is said to have a *sender* and a *receiver,* recognizing that to support a flow of data some packets (typically acknowledgments) will be sourced at the receiver and destined for the sender. A connection is said to have a *listener* and an *initiator* and a service is said to have a *server* and a *user.* It is very useful to treat these as orthogonal descriptors of the participants in a communication. Of course, a server is usually a listener and the source of data-bearing packets is usually the sender.

7.2 EFTP

The first 16 bits of all Ethernet packets contain its interface-interpretable destination and source station addresses, a byte each, in that order (see Figure 2). By software convention, the second 16 bits of all Ethernet packets contain the packet type. Different protocols use disjoint sets of packet types. The EFTP uses 5 packet types: data, ack, abort, end, and endreply. Following the 16-bit type word of an EFTP packet are 16 bits of sequence number, 16 bits of length, optionally some 16-bit data words, and finally a 16-bit software checksum word (see Figure 4). The Ethernet's hardware checksum is present only on the Ether and is not counted at this level of protocol.

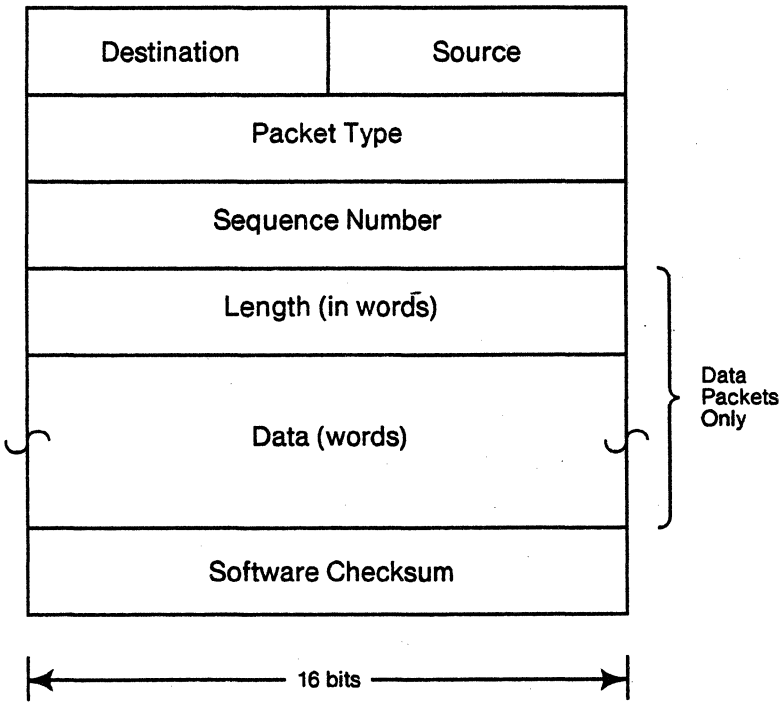


Fig. 4. EFTP packet layout.

It should be obvious that little care has been taken to cram certain fields into just the right number of bits. The emphasis here is on simplicity and ease of programming. Despite this disclaimer, we do feel that it is more advisable to err on the side of spacious fields; try as you may, one field or another will always turn out to be too small.

The software checksum word is used to lower the probability of an undetected error. It serves not only as a backup for the experimental Ethernet's serial hardware 16-bit cyclic redundancy checksum (in Figure 2), but also for protection against failures in parallel data paths within stations which are not checked by the CRC. The checksum used by the EFTP is a 1's complement add and cycle over the entire packet, including header and content data. The checksum can be ignored at the user's peril at either end; the sender may put all 1's (an impossible value) into the checksum word to indicate to the receiver that no checksum was computed.

7.2.1 Data Transfer. The 16-bit words of a file are carried from sending station to receiving station in data packets consecutively numbered from 0. Each data packet is retransmitted periodically by the sender until an ack packet with a matching sequence number is returned from the receiver. The receiver ignores all damaged packets, packets from a station other than the sender, and packets whose sequence number does not match either the expected one or the one preceding. When a packet has the expected sequence number, the packet is acked, its data is accepted as part of the file, and the sequence number is incremented. When a packet arrives with a sequence number one less than that expected, it is acknowledged and discarded; the presumption is that its ack was lost and needs retransmission [Metcalfe, 1973b].

7.2.2 End. When all the data has been transmitted, an end packet is sent with the next consecutive sequence number and then the sender waits for a matching endreply. Having accepted an end packet in sequence, the data receiver responds with a matching endreply and then *dallys* for some reasonably long period of time (10 seconds). Upon getting the endreply, the sending station transmits an echoing endreply and is free to go off with the assurance that the file has been transferred successfully. The dallying receiver then gets the echoed endreply and it too goes off assured.

The comparatively complex end-dally sequence is intended to make it practically certain that the sender and receiver of a file will agree on whether the file has been transmitted correctly. If the end packet is lost, the data sender simply retransmits it as it would any packet with an overdue acknowledgement. If the endreply from the data receiver is lost, the data sender will time out in the same way and retransmit the end packet which will in turn be acknowledged by the dallying receiver. If the echoed endreply is lost, the dallying receiver will be inconvenienced having to wait for it, but when it has timed out, the receiver can nevertheless be assured of successful transfer of the file because the end packet has been received.

At any time during all of this, either side is free to decide communication has failed and just give up; it is considered polite to send an abort packet to end the communication promptly in the

event of, say, a user-initiated abort or a file system error.

7.2.3 EFTP Shortcomings. The EFTP has been very useful, but its shortcomings are many. First, the protocol provides only for file transfer from station to station in a single network and specifically not from process to process within stations either on the same network or through a gateway. Second, process rendezvous is degenerate in that there are no mechanisms for finding processes by name or for convenient handling of multiple users by a single server. Third, there is no real flow control. If data arrives at a receiver unable to accept it into its buffers, the data can simply be thrown away with complete assurance that it will be retransmitted eventually. There is no way for a receiver to quench the flow of such wasted transmissions or to expedite retransmission. Fourth, data is transmitted in integral numbers of 16-bit words belonging to unnamed files and thus the EFTP is either terribly restrictive or demands some nested file transfer formats internal to its data words. And fifth, functional generality is lost because the receiver is also the listener and server.

8. Conclusion

Our experience with an operating Ethernet leads us to conclude that our emphasis on distributed control was well placed. By keeping the shared components of the communication system to a minimum and passive, we have achieved a very high level of reliability. Installation and maintenance of our experimental Ethernet has been more than satisfactory. The flexibility of station interconnection provided by broadcast packet switching has encouraged the development of numerous computer networking and multiprocessing applications.

9. Acknowledgements

Our colleagues at the Xerox Palo Alto Research Center, especially Tat C. Lam, Butler W. Lampson, John F. Shoch, and Charles P. Thacker, have contributed in many ways to the evolution of Ethernet ideas and to the construction of the experimental system without which such ideas would be just so much speculation.

10. References

[Abramson, 1970]
N. Abramson, "The Aloha System", *AFIPS Conference Proceedings*, vol. 37, Fall 1970.

[Abramson, 1975]
N. Abramson, F.F. Kuo, *Computer-Communication Networks*, Prentice-Hall, 1975.

[Ashenhurst, 1975]
R.L. Ashenhurst, R.H. Vonderohe, "A Hierarchical Network", *Datamation*, February 1975.

[Baran, 1964]
P. Baran, *On Distributed Communications*, Rand Corporation Memo RM-3420-PR, August 1964.

[Barnes, 1968]
G.H. Barnes, R.M. Brown, M. Kato, D.J. Kuck, D.L. Slotnick, R.A. Stokes, "The Illiac IV Computer", *IEEE Transactions*, C-17, vol. 8, August 1968.

[Binder, 1975]
R. Binder, N. Abramson, F. Kuo, A. Okinaka, D. Wax, "Aloha Packet Broadcasting—A Retrospect", *Proceedings of the National Computer Conference*, May 1975.

[Cerf, 1974]
V.G. Cerf, R.E. Kahn, "A Protocol for Packet Network Intercommunication", *IEEE Transactions on Communications*, vol. COM-22, no. 5, May 1974.

[Computer, 1974a]
"The Shrinking World: Computer Networks and Communications", *Computer*, IEEE Computer Society, February 1974.

[Computer, 1974b]
"Distributed-Function Computer Architectures", *Computer*, IEEE Computer Society, March 1974.

[Crocker, 1972]
S.D. Crocker, J.F. Heafner, R.M. Metcalfe, and J.B. Postel, "Function-Oriented Protocols for the ARPA Computer Network", *AFIPS Conference Proceedings*, vol. 40, May 1972. Reprinted in *Advances in Computer Communications*, edited by W.W. Chu, Artech House Inc., 1974. Reprinted in *Computer Communications*, edited by P.E. Green and R.W. Lucky, IEEE press, 1975.

[Crowther, 1975]
W.R. Crowther, F.E. Heart, A.A. McKenzie, J.M. McQuillan, and D.C. Walden, "Issues in Packet-Switching Network Design", *Proceedings of the National Computer Conference*, May 1975.

[Farber, 1973]
D.J. Farber, et al, "The Distributed Computing System", *Proceedings of the 7th Annual IEEE Computer Society International Conference*, February 1973.

[Farber, 1975]
D.J. Farber, "A Ring Network", *Datamation*, February 1975.

[Fraser, 1975]
A.G. Fraser, "A Virtual Channel Network", *Datamation*, February 1975.

[Heart, 1970]
F.E. Heart, R.E. Kahn, S.M. Ornstein, W.R. Crowther, D.C. Walden, "The Interface Message Processor for the Arpa Computer Network", *AFIPS Conference Proceedings*, vol. 36, May 1970.

[Heart, 1973]
F.E. Heart, S.M. Ornstein, W.R. Crowther, and W.B. Barker, "A New Minicomputer-Multiprocessor for the Arpa Network", *AFIPS Conference Proceedings*, vol. 42, June 1973. Reprinted in *Advances in Computer Communications*, edited by W.W. Chu, Artech House Inc., 1974.

[Kahn, 1975]
R.R. Kahn, "The Organization of Computer Resources into a Packet Radio Network", *Proceedings of the National Computer Conference*, May 1975.

[Metcalfe, 1972a]
R.M. Metcalfe, "Strategies for Interprocess Communication in a Distributed Computing System", *Proceedings of the Symposium on Computer-Communications Networks and Teletraffic*, Polytechnic Press, New York, 1972.

[Metcalfe, 1972b]
R.M. Metcalfe, "Strategies for Operating Systems in Computer Networks", *Proceedings of the ACM National Conference*, August 1972.

[Metcalfe, 1973a]
R.M. Metcalfe, "Steady-State Analysis of a Slotted and Controlled Aloha System with Blocking", *Proceedings of the Sixth Hawaii Conference on System Sciences*, January 1973. Reprinted in the *Sigcomm Review*, January 1975.

[Metcalfe, 1973b]
R.M. Metcalfe, *Packet Communication*, Harvard PhD Thesis, Massachusetts Institute of Technology Project Mac TR-114, December 1973.

[Metcalfe, 1974]
R.M. Metcalfe, "Distributed Algorithms for a Broadcast Queue", talk given at Stanford University in November 1974 and at the University of California at Berkeley in Febraury 1975, paper in preparation.

[Murthy, 1975]
P. Murthy, "Analysis of a Carrier-Sense Random-Access System with Random Packet Lengths", Aloha System Technical Report B75-17, University of Hawaii, May 1975.

[Ornstein, 1975]
S.M. Ornstein, W.R. Crowther, M.F. Kraley, R.D. Bressler, A. Michel, and F.E. Heart, "Pluribus—A Reliable Multiprocessor", *Proceedings of the National Computer Conference*, May 1975.

[Retz, 1975]
D.L. Retz, "Operating System Design Considerations for the Packet Switching Environment", *Proceedings of the National Computer Conference*, May 1975.

[Roberts, 1970]
L. Roberts, B. Wessler, "Computer Network Development to Achieve Resource Sharing", *AFIPS Conference Proceedings*, vol. 36, May 1970.

[Roberts, 1973]
L.G. Roberts, "Capture Effects on Aloha Channels", *Proceedings of the Sixth Hawaii Conference on System Sciences*, January 1973.

[Rowe, 1975]
L.A. Rowe, "The Distributed Computing Operating System", Technical Report Number 66, Department of Information and Computer Science, University of California, Irvine, June 1975.

[Rustin, 1972]
R. Rustin, Editor, *Computer Networks*, Proceedings of the Courant Computer Science Symposium 3, December 1970, Prentice-Hall Series in Automatic Computation, 1972.

[SDLC, 1974]
IBM Synchronous Data Link Control—General Information, IBM Systems Development Division, Publications Center, Department E01, P.O. Box 12195, Research Triangle Park, North Carolina 27709, 1974.

[SNA, 1975]
IBM System Network Architecture—General Infromation, IBM Systems Development Division, Publications Center, Department E01, P.O. Box 12195, Research Triangle Park, North Carolina 27709, 1975.

[Thomas, 1973]
R.H. Thomas, "A Resource Sharing Executive for the Arpanet", *AFIPS Conference Proceedings*, October 1973.

[Thornton, 1970]
J.E. Thornton, *Design of a Computer: the Control Data 6600*, Scott Foresman and Company, 1970.

[Walden, 1972]
D.C. Walden, "A System for Interprocess Communication in a Resource Sharing Computer Network", *Communications of the ACM*, vol. 15, no. 4, April 1972.

[Willard, 1973]
D.G. Willard, "Mitrix: A Sophisticated Digital Cable Communications System", *Proceedings of the National Telecommunications Conference*, November 1973.

[Wulf, 1972]
W. Wulf, R. Levin, "C.mmp—A Multi-Mini-Processor", *AFIPS Conference Proceedings*, Fall 1972.

Practical Considerations in Ethernet Local Network Design

by Ronald C. Crane and Edward A. Taft

October 1979, revised February 1980, reprinted October 1982

© Copyright Xerox Corporation 1980. All rights reserved.

Abstract: The design of a useful local-area communications network requires careful attention to a number of practical issues in addition to the theoretical concerns and analytical models emphasized in existing literature.

This report considers the design of the Ethernet local network, a broadcast multi-access bus system using carrier sense and collision detection. The design must be measured against the goals of the overall communications architecture which it supports, so the important properties of this architecture are briefly presented. The Ethernet design is then described in some detail, with emphasis on construction of the channel, interfaces, and low-level protocols. The report concludes with a summary of experience with an operational system and some suggestions for further improvement.

A condensed version of this paper was presented at the *Hawaii International Conference on System Sciences*, January 1980.

CR categories: 3.81, 6.35.

Key words and phrases: Local-area network, carrier-sense multi-access with collision detection (CSMA/CD), communications interface.

XEROX

Xerox Corporation
Palo Alto Research Centers
3333 Coyote Hill Road
Palo Alto, California 94304

1. Introduction

This report considers the design and implementation of the "Ethernet" local-area communications network. This is a passive bus-oriented system, with distributed control based on carrier-sense multi-access with collision detection (CSMA/CD). First described in [Metcalfe & Boggs, 1976], this architecture has been the subject of considerable analysis and has inspired numerous variations.

Very little, however, has been written on the actual implementation of Ethernet-style local networks. Past emphasis has been on the theory of operation and on modelling and analyzing the communications channel itself. But in a practical system, the choices made in the construction of the channel, interfaces, and low-level protocols are of crucial importance.

In this report, we begin in section 2 by presenting briefly the overall communications architecture that the Ethernet system is intended to support. Section 3 describes in some detail the prototype Ethernet implementation and its interface to a particular computer, and highlights some areas in which special care is required. Section 4 presents some operational considerations that are consequences of the present design. And in the final section we summarize our experience with an operational Ethernet system, discuss the strengths and weaknesses of the present design, and point out some areas requiring further work.

2. Overall design principles

The design of a local computer network must be considered in the context of the overall communications architecture it is intended to support. It is too frequently the case that communications systems are designed bottom-up and piecemeal, with results that are needlessly complex or are a poor match for the intended applications [McQuillan, 1979].

In the case of the Ethernet system, the context is an *internetwork* architecture called "Pup" [Boggs *et al*, 1980]. As in other such designs [Cerf & Kahn, 1974; Sunshine, 1977], it is the purpose of the Pup internet to facilitate computer-to-computer communication among diverse host systems interconnected by a heterogeneous collection of networks. These networks are coupled by *internetwork gateways*, which are hosts that are connected to more than one network and are willing to forward packets among them.

2.1 The computational and communications environment

The computing environment addressed by the Pup architecture consists primarily of "Alto" minicomputers [Thacker *et al.*, 1979] and other personal computers capable of high-quality interaction with human users in many application areas—text editing, illustration, document preparation, interpersonal communication, and others. These machines appear in clusters, sometimes with over 100 Altos in one *campus* (building or group of adjoining buildings). Each

campus is supported by various specialized servers providing remote access to shared resources such as large file systems, magnetic tapes, and high-quality printers. Efficient, high-bandwidth communication within a campus is essential.

The campuses are interconnected by slower communication facilities, consisting mostly of leased telephone circuits, but also including other networks such as the Arpanet and the ARPA Packet Radio network. These facilities must also support the full generality of internetwork communication, just with lower performance.

2.2 Pup principles

This section presents the highlights of the Pup internetwork architecture, in preparation for examining their impact on local network design. These topics are elaborated upon in [Boggs *et al.*, 1980].

Simplicity. A guiding principle of the Pup design has been the desire for simplicity, especially at the lower levels. The lowest level of internet protocol is a simple end-to-end datagram—a standardized packet (called a "Pup") which is media-independent but otherwise retains the common properties of the underlying packet networks. Maintaining simplicity in the low-level design facilitates introduction of new network technology and best accomodates new applications. Furthermore, the computational environment is dominated by relatively lightweight hosts requiring high-bandwidth intercommunication, so simplicity of the basic protocols is essential to efficient implementations.

End-to-end protocols. The basic facility provided by the Pup internet is the transport of independently addressed Pups from a source process to a destination process. All higher-level interprocess communication primitives, such as connections, streams, transactions, etc., are implemented by higher-level protocols that are strictly a matter of agreement among the communicating end processes. This arrangement makes the internet itself very simple, imposes minimal functional requirements on the underlying packet networks, and contributes to reliability by keeping fragile end-to-end state out of intermediate agents such as gateways.

Individual networks viewed as packet transport mechanisms. The only capability required of an underlying packet network is the ability to transport Pups from a source to a destination within that network. To this end, an *encapsulation* technique is defined for each type of network, permitting a Pup to be carried transparently. Depending on the requirements of the network, encapsulation may include various transformations such as adding headers and trailers, performing fragmentation, applying error correction or other coding techniques, and so on. The Pup specification has nothing to say about such transformations except that they be *inverted* as an encapsulated Pup leaves the network.

Reliability of transport. Transporting networks are *not* required to be perfect; rather, they should simply give their "best efforts" to deliver each Pup [Metcalfe, 1973]. Packets may be lost, duplicated, delivered out of order, after a great delay, and with hidden damage. For performance reasons, the underlying networks should be designed to make such events uncommon. But it is the responsibility of the end processes to attain the level of reliability *they* require, by use of suitable end-to-end protocols.

2.3 Consequences for local network design

In light of the principles of the Pup architecture, it is clear that the functional capability required of a local network is really quite simple: the transport of an encapsulated, individually addressed Pup between any two hosts connected to the network, with good but not necessarily perfect reliability. Other capabilities that might be engineered into a local network are of little use in the Pup design because they are generally difficult to extend into the full internet environment.

Given this relatively simple functional requirement, one can concentrate on other important aspects of a local network design. These include *availability*, the network's resistance to catastrophic failure (as opposed to isolated packet loss, which is of little consequence); *performance*, the extent to which the network's capacity is accessible to the end communicating processes; and *maintainability*, the cost and effort required for installation and maintenance.

These properties of a local network are of course heavily dependent on the theory that underlies the local network design. In the case of the Ethernet system, that basis is carrier-sense multi-access with collision detection (CSMA/CD). That technique has by now received fairly extensive theoretical analysis [Metcalfe & Boggs, 1976; Agrawala *et al.*, 1977; Almes & Lazowska, 1979; Tobagi & Hunt, 1979].

Equally important are a number of considerations that arise in the design and engineering of a functioning and useful local network. These considerations apply not only to the communications channel itself but also to the transceivers, controllers, and low-level software and protocols, as discussed in the next section.

3. Ethernet design and implementation

We now describe the existing Ethernet system: the shared channel, the interface to the Alto minicomputer (the predominant type of Ethernet host), and the low-level protocols. We start with a brief summary of system operation and then proceed with the component descriptions.

3.1 Theory of operation

In a CSMA/CD system, stations *contend* for use of the "Ether" (common broadcast communications channel) until one of them *acquires* and uses it to transmit a packet. *Carrier sense* is used to *defer* transmission until the Ether is quiet (no other carriers present). When quiet is detected, the source machine proceeds to transmit.

During transmission, the source listens for a *collision* (any other carrier on the Ether at the same time). Since other stations will defer once they see a carrier, the first round-trip propagation time is the only time that collisions can occur in the network. If no other carrier is detected within one-round trip time, the source is said to have *acquired* the Ether. If another carrier is detected, the packet is aborted and a *jam* signal is transmitted to ensure that the collision is also detected by all other participants in it. After jamming, each station waits a random length of time and tries again. Justification for these design choices and much more detail may be found in [Metcalfe & Boggs, 1976; Shoch, 1979].

3.2 Ethernet system overview

The prototype Ethernet system is a bus-oriented communications network providing a raw bandwidth of 3 megabits per second, connecting up to 255 computers, and spanning a linear distance of 1 kilometer. A piece of 75-ohm coaxial cable with terminators on each end constitutes the bus. The cable is typically located in the ceiling and travels over or near areas to be served by the network. The overall structure of this system is illustrated in figures 1 and 2.

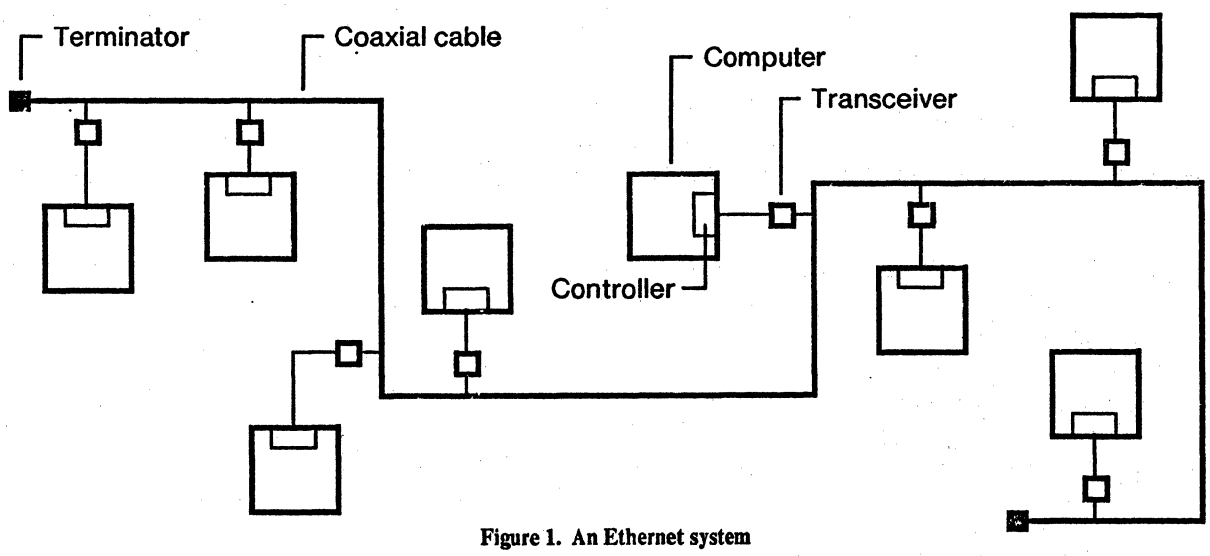


Figure 1. An Ethernet system

Transceivers are located next to the cable and are attached to it by pressure taps. The transceiver provides ground isolation and a high-impedance connection to the cable for transmitting and receiving. Twisted pair cable brings signals and power to the transceiver from a controller in

the computer. This portion of the Ethernet system is identical for all types of hosts.

The Ethernet controller for the Alto computer is implemented partially in hardware, partially in microcode, and partially in software. The hardware contains a small (16-word) buffer, parallel-serial and serial-parallel converters, CRC generator and checker, and Manchester phase encoder and decoder. The microcode performs address recognition, schedules retransmissions when collisions occur, and provides the interface seen by the software. The software is responsible for basic packet formatting, Pup encapsulation and decapsulation, and interfacing to the network-independent software, which in turn distributes and collects Pups (internetwork datagrams) to and from the end communicating processes.

Following is an elaboration of specific details of the system implementation.

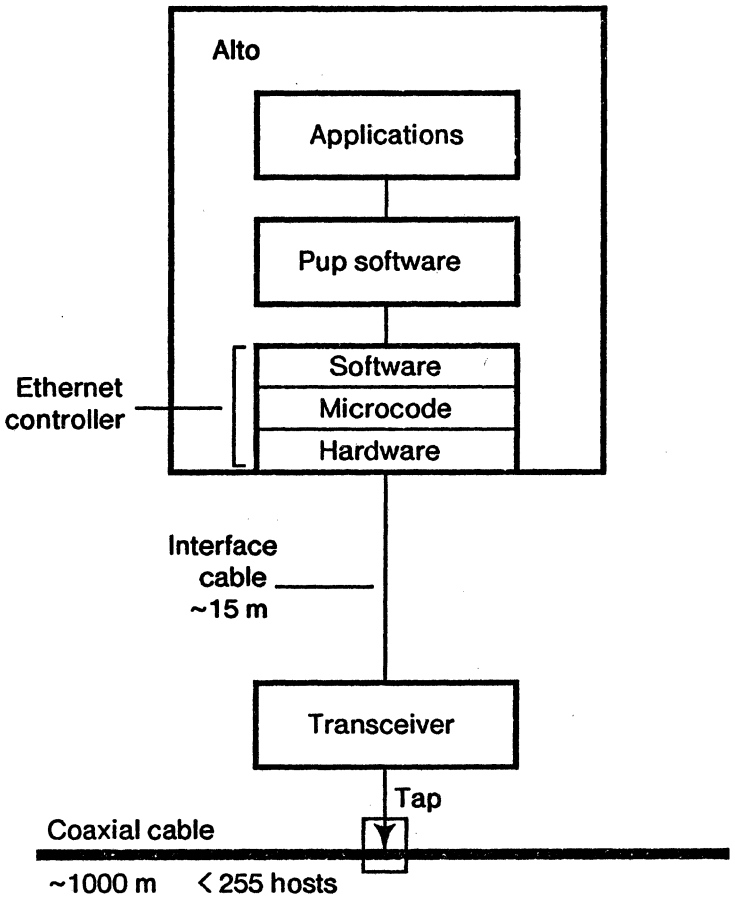


Figure 2. Alto-Ethernet connection

3.3 Cable and taps

Cable selection has been based on low signal loss and the ability to be tapped using existing cable TV technology. A foam type RG/11 coaxial cable is used for the Ether. Transceivers are connected to it using a cable tap manufactured for the cable television industry by Jerrold

Electronics. It comes in two parts: a split metal block with two pins which clamps the cable and pierces the shield, and a 3/8-inch diameter coaxial pin assembly (tap) which screws into a threaded hole in the block, pierces the core dielectric of the cable, and contacts the center conductor.

The common coaxial cable and the individual transceiver-to-controller cables are typically located in the space above the false ceilings in office buildings. While these cable systems utilize less than 30 volts and are current-limited, thereby exempting them from many building regulations, use of the space above hung ceilings as an air return duct (plenum) for the building ventilation system exposes the system to stringent fire regulations.

The critical parameters for approval of the cable are its ability to propagate a flame and the number of pounds of combustible material that it contains. Teflon requires a 95 percent oxygen atmosphere to burn, very high temperatures to cause decomposition, and has a relatively low energy content per pound. Teflon coaxial cable has recently received UL approval for use in air plenums [Du Pont, 1979].

3.4 Transceiver

The transceiver contains a small quantity of electronics to drive and receive signals on the cable, detect collisions, maintain ground isolation, and protect the shared channel from certain failures of the transceiver, controller, or host. Figure 3 illustrates the functions of the transceiver.

Signalling Conventions. Data on the cable is Manchester encoded. This coding has the property that it has a transition in every bit cell and has a 50 percent duty cycle. Bits are phase-encoded in the controller before being passed to the transceiver. The first half of a bit cell contains the complement of the bit and second half contains the bit itself. There is always a transition in the middle of the bit cell: a positive edge corresponds to a "one" bit and a negative edge to a "zero". The voltage levels transmitted into the cable are +3 volts for "on" and 0 volts for "off". Carrier is detected by the presence of transitions on the cable; when no transmitter is active, the channel is quiescent in the "off" state.

Collision detection. The transceiver contains a circuit that performs the function of collision detection. It compares the received data with that being transmitted and (after suitable filtering to eliminate spikes due to skew) produces a *collision* signal whenever there is a difference. (The host's controller monitors *collision* only while it is transmitting.) The circuit is located in the transceiver rather than the controller because that is the point at which the skew between transmitted and received signals is at a minimum.

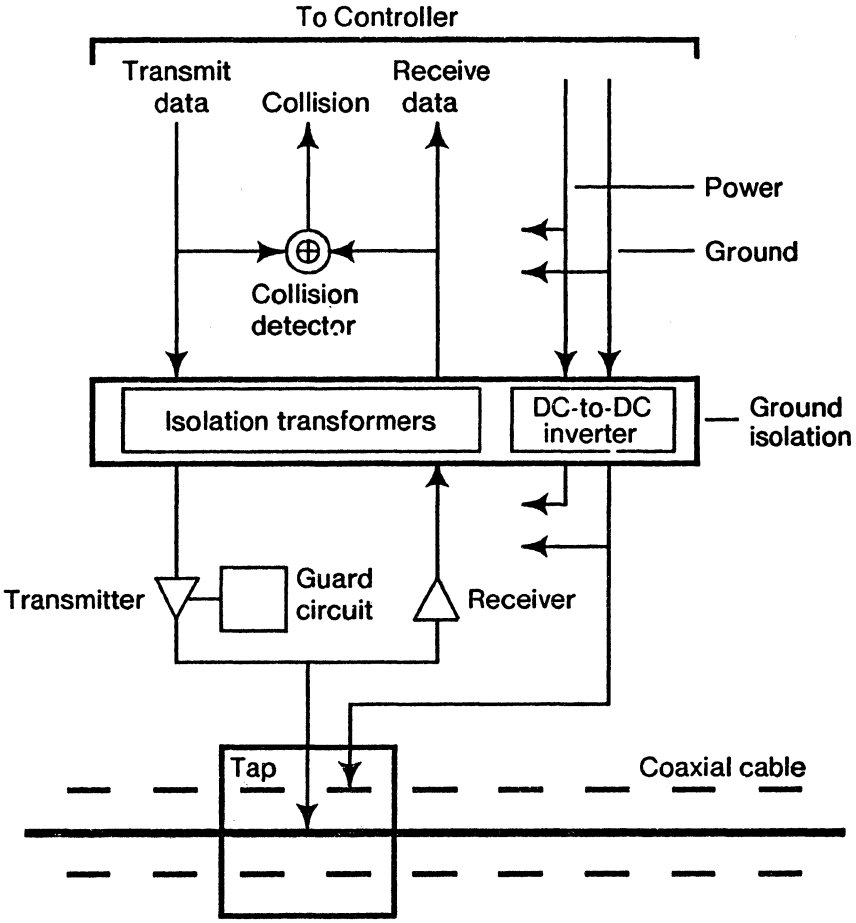


Figure 3. Ethernet Transceiver

This method of detecting collisions depends on the fact that with Manchester coding, the transmitter is off for half of each bit period. If some other host is transmitting simultaneously, any of its "on" intervals that overlap this transmitter's "off" intervals will be detected as collisions. An exclusive-OR gate is used in the Ethernet transceiver. This technique is limited to receivers using a fixed DC threshold for a reference. Under conditions of greater attenuation, higher data rate, or longer cables, an analog hybrid would be required to generate the collision signal.

Ground isolation. There are two ground references in the system. One is the common coaxial cable shield and the other is the local ground associated with each Alto or other host computer. It is important that these not be tied together, since one local ground may differ by several volts (60 Hz) from another local ground. Connection of several local grounds to the common cable could cause a large current to flow through the cable's shield. The ground isolation provided by the transceiver eliminates this connection through an Alto. It is equally important to prevent this connection via contact of a tap block with building grounds such as cable trays, conduits, or ceiling hangers. (Later we describe one consequence of failure to observe this isolation.)

Protection against failures. Each transceiver provides a high-impedance connection to the cable in both the power-on and power-off states. It contains a guard against transceiver circuit failure, which monitors the length of transmitted pulses and latches the unit off if they are too long. (This may be reset by removing power for several seconds.) Additionally, the AC coupling between the host computer and the transceiver prevents the channel from being permanently jammed by a controller whose *transmit data* signal becomes stuck on.

The transceiver electronics themselves are subject to damage from electrical transients in the cable. In more recent designs, transceivers have been modified to withstand 20 to 30 volts between the center conductor and shield.

RF emissions. Consideration of RF emissions is not only a courtesy to other electronic equipment and radios but is also necessary to meet FCC regulations. RF emissions may arise from the interface cable between controller and transciever, from the transceiver itself, and from the tap block connection. All of these sources can be reduced to acceptable levels.

The existing system uses unshielded twisted-pair between the transceiver and the controller, driven in a single-ended manner. A significant improvement results with use of balanced differential signalling and an overall shield.

Improper design of the transceiver isolation stage can cause a common-mode voltage to appear between its connection to the interface cable ground and its connection to the shield of the coaxial cable. The introduced signal may be at either the signal frequency or the switching frequency of the DC-to-DC inverter or both, depending on the design. These issues must be addressed at design time, as they are very difficult to correct after the unit is built.

It is important to minimize the impedance between the tap block ground and the cable shield, because a small voltage develops across this impedance whenever a transceiver is transmitting.

Signal symmetry. It is essential that the symmetry of the signal be preserved as it travels from a controller into a transceiver, through the coaxial cable, back into a transceiver, and into a receiving controller. Considerable care has been taken in the design of the cable drivers and receivers for both the interface cable and the common coaxial cable.

On the coaxial cable, transmit pulses are 3 volts in amplitude on the cable and coupled into the cable through a diode. The receiver is DC-coupled to the cable and has threshold of about 1.3 volts, which is the centerpoint of a waveform received 1 kilometer from the transmitter. Near the transmitter the threshold should be 1.5 volts, but there the rise and fall times of the signal are only 12 nanoseconds, so the threshold voltage is not as important nearby as it is at the end of the cable where the signal waveform looks more like a sine wave.

The second part of maintaining symmetry is the method of driving and terminating the interface cable. The present system uses open-collector drivers and a receiving end composed of a TTL gate and a termination impedance equal to the cable impedance and tied to a voltage twice the threshold voltage of the TTL gate (approximately 2.8 volts). This provides a symmetrical signal about the threshold and prevents reflections in the cable.

3.5 Controller

While controllers have been implemented for several computers, the Alto controller, whose hardware is diagrammed in figure 4, is the most widespread at present.

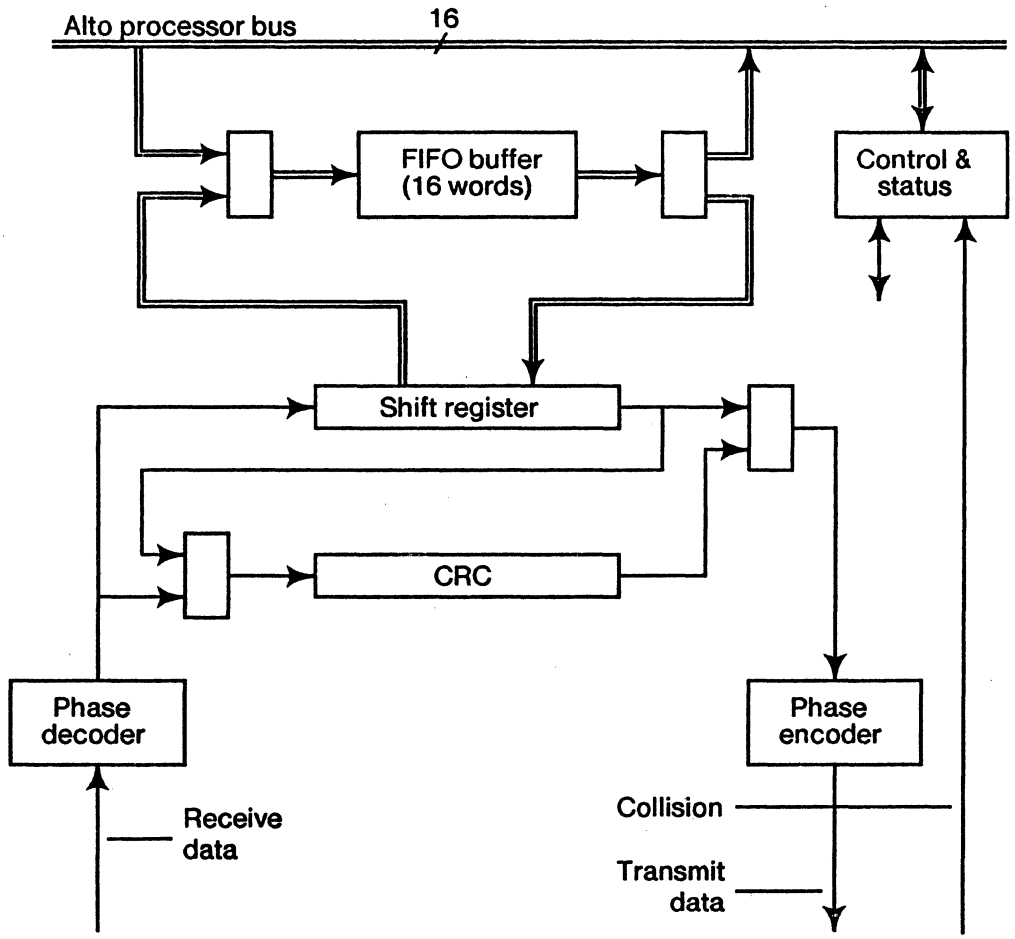


Figure 4. Alto-Ethernet controller hardware. Single lines are serial data paths. Double lines are 16-bit parallel data paths. This controller is half-duplex and is configured either to transmit or to receive at any given time. Newer designs have independent transmit and receive paths.

Controller hardware considerations. The size and complexity of a controller is dependent on the architecture and speed of the machine to which it is interfaced. The I/O architecture determines the amount of hardware needed to make a connection to the machine, and the machine's speed determines whether some of the functions could be performed by microcode or software instead of hardwired logic. In the case of the Alto, the hardware is quite simple because the Alto's microprogrammable processor controls I/O devices directly, and data transfers, address filtering, and exception handling are performed by microcode. (For an elaboration on this, see [Thacker *et al.*, 1979].)

The controller must provide adequate internal buffering of bits to accomodate the bandwidth and latency of the host computer's I/O system. Since high-speed buffer chips are expensive, the smallest amount of buffering that meets this requirement is used. In all of the systems we have built so far, the I/O bandwidth of the processor exceeds the bit rate of the channel and has a service latency time considerably less than a packet transmission time. Thus, buffers may be small—16 words in the Alto. In an interface to a processor which has a low I/O bandwidth, the buffer would have to be able to hold an entire packet, and additionally the packet rate to that machine would have to be sufficiently slow to allow the machine to empty the buffer before the next packet arrived.

It is possible for the host to lose incoming packets if the controller fails to listen to the Ether all the time. This is most prevalent in half-duplex controllers (such as the one used in the Alto) where there is significant turnaround time between ending a local transmission and restarting the receiver. The maximum turnaround delay without packet loss determines the minimum allowable packet separation on the cable.

In the prototype Ethernet system, minimum interpacket separation is only a few bit times. In the Alto controller, there is a microcode/software turnaround delay considerably larger than this; consequently, the Alto does miss some incoming packets, with performance ramifications that will be discussed later (section 5.1). This is discussed at greater length in [Shoch, 1979].

A design in which the hardware automatically switches modes at the completion of transmission eliminates this problem, provided that the microcode and software are prepared to deal with an incoming packet right on the heels of one just transmitted. A full-duplex interface, with independent transmit and receive data paths, is structurally simpler (though requiring slightly more hardware); it has the added advantage of enabling the host to receive its own transmissions, which is useful for self-testing and diagnosis. Most of our newer designs are full-duplex.

Phase decoding techniques. Each controller contains a circuit that recovers data and clock from the Manchester encoded signal received from the cable. Two techniques are in use in the current Ethernet system. Both have the property that they can synchronize to the one-bit preamble of the packet and successfully receive the rest of the packet. The first type, used in the Alto, is analog in nature, using two one-shots to recover data and clock and to generate a *carrier detect* signal. The second type, used in newer machines, is digital in nature, using a finite state machine that samples the received signal at eight times the bit rate. The state machine also checks the timing between transitions, flagging malformed signals as probable collisions.

Collision handling. Here we discuss the details of contention in the network. As described previously, a transmitting station detects a collision, transmits a jam pulse, and retransmits after waiting a random interval of time. To ensure fairness, it is important that all stations use the same algorithm.

When the *collision* signal is raised by the transceiver, the controller ceases data transmission immediately and turns the transmitter continuously "on" for 3 microseconds, then shuts down. This *jam* signal ensures that all participants in the collision are made aware of it and also shut down.

When a collision occurs, a retransmission timer is started. After it expires, the controller may retransmit the packet (of course, first deferring to any ongoing transmissions in the usual manner). The timer is set to a random value computed using a *binary exponential backoff* algorithm. The value chosen is uniformly distributed between zero and $2^n - 1$, where $n = \min(retries, 8)$ and *retries* is the number of previous unsuccessful attempts to transmit the packet. The unit of time is 38 microseconds, which—in theory—is chosen to approximate the maximum contention interval or *slot* time [Metcalfe & Boggs, 1976], but in practice is simply any convenient quantum that exceeds the slot time.

The Alto implementation of the binary exponential backoff algorithm is very simple. For each transmission attempt, a mask, initialized to zero before the first attempt, is ANDed with a suitable source of 8-bit pseudo-random numbers (e.g., 8 rapidly-changing bits of an unsynchronized real-time clock) to produce the timer value. The mask is then left-shifted one bit and a "one" introduced into the low-order bit. This process repeats for each retransmission attempt until the packet is successfully transmitted. If 16 successive collisions occur, the controller abandons attempting to transmit the current packet; this *load overflow* condition is an indication of extreme overload or network failure and is left for higher-level software to deal with.

Receivers should take some care to detect and filter out packets that have suffered collisions, and to do so at a fairly low level so as not to burden higher-level software with the expense of dealing with these relatively common events. There are several ways to do this. The design parameters of the present Ethernet system (1 kilometer, 3 megabits per second) are such that collisions give rise to packets less than 32 bits long, so the controller can simply filter out such "runt packets". An additional level of redundancy is afforded by use of a 16-bit cyclical redundancy check (CRC) and by rejecting packets that do not end on a 16-bit word boundary. Newer controller designs that employ digital phase decoding can detect encoding violations, which are frequently an indication of collision.

Minimizing the probability of collision. A collision occurs when a transmitter fails to defer to another transmission already in progress. The failure is due to the time delay between start of packet transmission in one machine and the carrier sense ("decision to transmit") circuitry in the other machines. The worst case of this determines the *acquisition time* of the Ether. Most of this time is due to propagation delay through the cable, but some is due to transceiver and controller delays.

For any given packet, the acquisition time is given by the following:

$$T_a = 2\,T_p + 2\,T_d + T_s$$

where T_p is the propagation delay through the cable to the other machine; T_d is the delay through the transceiver, transceiver cable, and controller up to decision gate; and T_s is the delay from the decision to start transmission until the appearance of the start of the packet on the common cable.

The probability of a given packet colliding with any packet from some other machine transmitting n packets per second is:

$$P = n\,T_a$$

assuming $n \leq 1/T_a$ and packet transmission time $\geq T_a$. (If packet transmission time is less than T_a then the concept of acquisition breaks down, as does the fundamental concept of transmitter-based collision detection. This is the point at which Ethernet channel behavior moves toward that of a slotted Aloha channel [Abramson, 1977].)

The consequence of this with respect to controller design is that the decision to transmit should be made just before the packet would actually be sent to the cable. In particular, the timer that enforces the minimum spacing between packets should always start when an end of packet is detected (loss of carrier) instead of starting after the decision to transmit. Thus if the Ether has been quiet for a while, a transmission proceeds immediately rather than waiting for the timeout.

3.6 Low-level protocols

In the Pup architecture, the purpose of a low-level protocol is to deal with peculiarities of the network. Functions in excess of this (e.g., connections, flow control, sequencing, perfectly reliable delivery) don't belong here but rather at higher levels of protocol—which, in the Pup design, are all end-to-end. In the Ethernet system, an attempt has been made to provide the simplest possible mechanisms that have the required functionality and performance.

Packet format. The basic Ethernet packet format is shown in figure 5. A packet begins with a *sync bit*, which is always a one and whose leading edge enables the receiver to acquire bit phase. Following this are two 8-bit address fields, the *destination host* and the *source host*; these are followed by a 16-bit *packet type*, zero or more 16-bit words of *data*, and a 16-bit *cyclical redundancy check* (CRC). The sync bit and the CRC are generated and stripped off by the controller hardware; all remaining information is transferred to or from the host computer's memory.

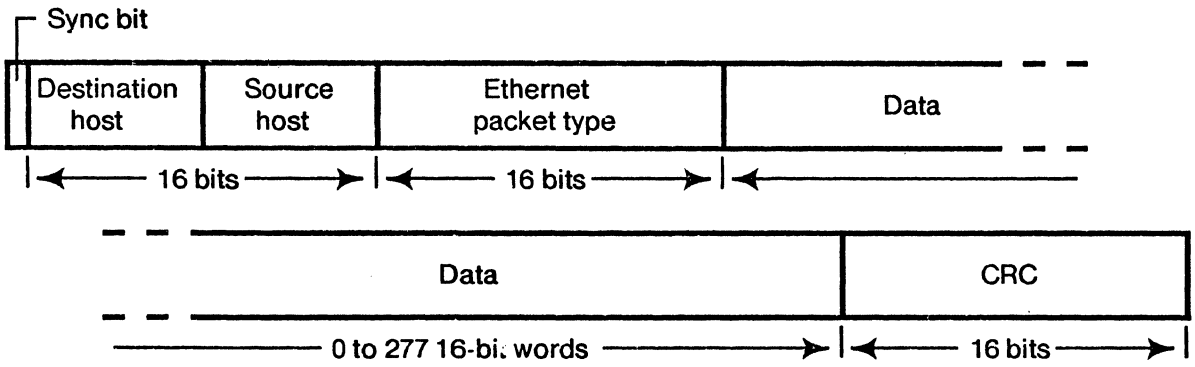


Figure 5. Ethernet packet format. Sync bit and CRC are added and removed by controller hardware. All other fields are generated and consumed by software, though the destination host is also examined by receiver microcode to perform address filtering. Maximum packet length is a software convention.

Addressing. The addressing conventions used in the prototype Ethernet system are very simple. Each host is assigned a unique 8-bit address that is readable by software running in the host, and each packet transmitted is labelled with its source and intended destination address.

Of course, the Pup internet protocol requires an address space much larger than this, extended both upward (to encompass multiple networks) and downward (to address multiple processes within a machine). Those addresses, however, are in the Pup itself—that is, encapsulated within the *data* portion of an Ethernet packet.

Received packets are copied into memory under control of an *address filter*, which in the Alto is implemented in microcode. Upon receiving the first word of a packet, the microcode compares the destination host field against an address supplied by software (normally the host address assigned to the machine), and copies the packet into memory only if they are equal.

The software can also set the address filter to be *promiscuous*, meaning that every received packet is to be copied into memory. This is useful for network monitoring, protocol debugging, and other purposes.

A destination host address of zero is used to indicate a *broadcast* packet, which every active host should copy into memory. Broadcasts are useful for locating nearby resources (on the same network) and for distributing information of general interest. But broadcasts must be used with discretion, since all hosts that receive one incur a certain amount of software overhead, if only to discard it.

A generalization of broadcast is *multicast*, in which a packet may be addressed to a predefined set of hosts, referred to as a *multicast group* and identified by a *multicast ID*. In the present Ethernet system, a form of multicast may be implemented by reserving certain addresses as multicast IDs rather than host addresses. A host then joins a multicast group by setting its address filter to accept packets addressed to the group's ID. (This simple scheme has some obvious limitations, such as the inability for a host to belong to more than one multicast group simultaneously. But the address filtering may be made arbitrarily elaborate by loading different microcode, or by setting the microcode address filter to *promiscuous* and doing address filtering in

the software.)

While broadcast and multicast are easy to implement within a single network, they are difficult to extend into the full Pup internetwork environment. Consequently, existing Ethernet systems make little use of these techniques (multicast especially), and we have intentionally avoided introducing added complexity into controllers or low-level protocols to provide for them.

Other conventions. The *packet type* field is reserved, by software convention only, to identify the specific higher-level protocol under which the *data* should be interpreted. While Pup is the predominant higher-level protocol at present (and hence most Ethernet packets identify their contents as being Pups), there do exist other protocol architectures—notably the ARPA Internet—whose basic packets may also be encapsulated within Ethernet packets. Also, there are a few applications that are highly specialized to the Ethernet or to the Alto, such as the initial step of bootstrap loading; these utilize their own Ethernet protocols rather than being based on Pup.

The Ethernet system does not employ low-level acknowledgments. Packet errors are so infrequent that low-level acknowledgments are not needed: end-to-end protocols can easily cope with the network's error rate. Likewise, no special low-level protocols are required to gain access to the network, since the channel is controlled strictly by contention.

4. Operational considerations

This section briefly highlights a few of the important operational characteristics of the present Ethernet design.

4.1 Installation

Simple rules describe the installation of the coaxial cable. It must have a terminator at each end, it must not branch, and it must never be connected to a building ground (by contact to connector casings or tap blocks). The terminators should be located in accessible locations so that a time domain reflectometer can be conveniently connected to the end for checkout. The cable should be broken every 100 meters or so and a connector installed in an accessible location; this facilitates installation of the cable and is also useful in fault isolation, as will be described.

Installation of a tap and transceiver is relatively straightforward and does not require taking the network down. The tap block is placed on the cable and tightened, piercing the jacket and making contact with the shield. The jacket and shield coring tool is screwed in and removed, leaving a hole for the tap. The transceiver with tap is screwed into the tap block, completing the job. The tap and tap block may subsequently be removed and the hole taped over with no effect on the rest of the system.

It is best to install taps at least a few feet apart so as to distribute the taps' and transceivers' capacitance and impedance along the cable rather than lumping it all in one place. In the prototype Ethernet system, running at 3 megabits per second, this is not especially important (one existing Ethernet cable, about 500 meters long and with over 120 hosts, has several clumps of closely-spaced transceivers with no ill effects); however, at higher data rates this would become a more serious concern.

4.2 System configuration

The current Ethernet design uses 8-bit host IDs assigned on a per-network basis. If a machine is moved from one network to another—a fairly common event for Altos—its ID may have to be changed to fit into the address space of the new network. This is an operational annoyance.

An alternate scheme is to provide each host with a larger ID (at least 32 bits) which is unique across all networks. This simplifies the movement of machines from one network to another. Of course, some central source of IDs must be maintained to assure that none are repeated.

Aside from assignment of host IDs, no central coordination is required when a host is added to or removed from a network.

4.3 Reliability and fault isolation

There are two classes of failures: those that disable a single host and those that take down the entire network. Because the common channel is passive, most failures of transceivers and interfaces affect only one host, and it is immediately obvious which host it is.

Network failures can be caused by a transceiver stuck in the "on" state, a shorted transceiver, or a missing terminator. All of these have the effect of reducing the signal to noise ratio on the cable below an acceptable level.

A time domain reflectometer (TDR) has proven valuable in locating shorts, missing terminators, and even transceivers in the "on" state (since they have a low output impedance while transmitting). It is useful to have a reasonably accurate record of how the cable is routed, so that the linear distance from the TDR may be used to determine the approximate location of the failing component.

Segmenting the network is a useful technique when a TDR is not available, assuming that connectors have been installed in the cable at reasonable intervals. The network is broken into segments and each segment terminated at the point of separation. The segment that does not work contains the fault. (Multiple Ethernet systems interconnected by gateways have the property that a failure is localized to one network.)

5. Experience with an operational design

The prototype Ethernet system has served for several years as the dominant local network in a nationwide Pup internet used regularly within a number of Xerox research and development communities. The internet presently consists of about 25 Ethernet systems interconnected by 20 or so gateways and various long-haul transmission facilities. Over 1000 hosts (mostly Altos and other personal minicomputers) are connected to these networks.

Practical experience with the Ethernet system has shown the strengths and weaknesses of the present design. This section summarizes our experience and highlights those aspects of the design that could be improved.

5.1 Performance

The analytical model of CSMA/CD performance under *heavy load* and *overload* conditions, first presented in [Metcalfe & Boggs, 1976], has now been verified experimentally [Shoch & Hupp, 1979]. The network remains stable, its capacity is shared fairly among the contending hosts, and efficiency exceeds 97 per cent when traffic consists of relatively large packets (about 4000 bits).

Of at least equal importance is the performance of higher-level applications under *typical* conditions. Practical experience has shown that the consequences of certain design and engineering choices, such as those presented previously, totally dominate the undesirable effects predicted by analysis of an overloaded Ethernet system. (For example, in [Almes & Lazowska, 1979] it has been shown that the "binary exponential backoff" algorithm used for contention resolution in the Ethernet gives rise to a rather curious last-in-first-out acquisition of the channel. This phenomenon has indeed been observed under extreme overload conditions, but is of no consequence during normal operation.)

Typical conditions. It is useful to characterize what "typical" conditions are. Extensive measurements have been performed on one operational Ethernet system to which approximately 120 computers are connected. These machines include personal computers, servers, time-sharing systems, and gateways to other parts of the Pup internet. Users of these machines use the network regularly for file transfer, electronic mail, bootstrap loading, software distribution, and many other purposes.

In a typical 24-hour period, 2.2 million packets are carried in this network, totalling approximately 300 million bytes. For a 3-megabit per second channel, this amounts to only 0.9 percent utilization of the available bandwidth. But the utilization has high variance, and observed utilization has been as high as 3.6 percent for one hour, 17 percent for one minute, and 37 percent for one second. On the average, only 0.79 percent of all packets are delayed due to deference, and less than 0.03 percent are involved in collisions.

Thus, the existing system could support a substantial increase in traffic without a noticeable increase in latency. This is as it should be. While it is reassuring to know that the Ethernet system will remain stable under overload conditions, it would be bad practice to operate a local network regularly at more than a fraction of its peak capacity, especially in view of the high variations in utilization.

End-to-end performance. In the environment served by Pup, the most important metric for evaluating the Ethernet system is the performance delivered to the communicating end processes. In view of the negligible impact on overall performance due to contention for the shared channel, this performance is primarily a function of other components of the system. In the prototype implementation, there are two principal limitations on end-to-end performance.

Receiver turnaround. The inability of an Alto interface to receive consecutive packets or to receive a packet that immediately follows that Alto's own transmission has serious performance consequences. Many packets are missed simply because the receiver is not turned on at the time a packet begins to arrive. This occurs most frequently in servers and gateways, which may be communicating with several hosts at the same time; but it can also occur in any pair of hosts that happen to be sending to each other simultaneously.

Timeout-driven higher-level protocols can and do cope with lost packets, but when the rate of loss becomes too great the end-to-end throughput drops dramatically. This effect completely swamps the degradation caused by contention and damaged packets. For most applications this is still not noticeable, but for some it is serious enough that we have found it necessary to tune higher-level protocol implementations so as to reduce the probability of losing packets due to turnaround latency. This is a serious intrusion upon what is supposed to be a network-independent protocol.

This problem is due to the Alto controller hardware, which is half-duplex and requires restart by software or microcode after transmission or reception of a packet. The present design was arrived at after consideration of maximum chip count (approximately 75 TTL MSI ICs, using technology available in 1973). New designs should be capable of receiving consecutive packets. Fortunately, advances in IC technology have now made this possible at low cost.

Software overhead. End-to-end throughput is primarily limited by per-packet processing overhead in the software. For example, present implementations of the Pup Byte Stream Protocol, written in a high-level language (BCPL or Mesa) and running on the Alto, are capable of maintaining a sequenced, flow-controlled data stream at the rate of about 600 kilobits per second, at which point both sending and receiving computers are CPU-bound. While this performance is fairly respectable considering the modest processing capability of an Alto, it is far below the capacity of the Ethernet channel.

Reducing per-packet software overhead may be accomplished by designing "smart" controllers that are better tuned to the needs of higher-level protocols. Sources of overhead include packet completion interrupt handling, Pup encapsulation and decapsulation, collection of packets from and delivery to the correct processes, and higher-level protocol overhead such as sequencing, generation and processing of acknowledgments, and so on. Some of these tasks could be taken over by Alto microcode or (in other designs) by a dedicated front-end microcomputer.

The danger in this approach is that higher-level protocol functions may become inappropriately embedded in the controller, thereby blurring the levels of protocol and reducing the flexibility available to the end communicating processes. To avoid this, one should attempt to provide such functions in the form of operations *accessible from software* as opposed to ones *permanently interposed* between the network and the end processes. (This observation applies equally well to other operations traditionally thought of as being functions of a communications interface—for example, encryption [Needham & Schroeder, 1978].)

5.2 Reliability

One of the major design goals of the Ethernet system was that it be extremely reliable, in the sense that the shared communications channel should be immune from catastrophic failure. Failures that affect an individual host or that cause isolated packets to be lost or damaged are of secondary importance. This goal (among others) led to the passive bus structure in which most of the components—in particular, all the active ones—are associated with the host computers rather than being part of the channel itself, thereby reducing the likelihood that a single malfunction will cause total failure of the network.

This approach has been very successful in practice. Failures of an entire Ethernet system are quite rare; for example, the heavily-populated network described previously (120 hosts) suffers outages only a few times per year, and only for a few minutes at a time while the problem is located and corrected. Failures that affect individual hosts or that increase the packet error rate are, of course, more frequent. It is worth highlighting some of the most common malfunctions.

Total failures. The usual causes of catastrophic failure are incorrectly installed taps, shorted or malfunctioning transceivers, and broken controllers that transmit data indefinitely.

A bad tap or shorted transceiver is potentially the most troublesome problem to deal with, since locating and correcting it might require inspecting long stretches of the coaxial cable. (This concern has motivated network structures such as the "star-shaped ring", which features centralized fault isolation [Saltzer & Pogran, 1979].) In practice, however, such failures are almost invariably associated with physical activity such as installing a new tap or transceiver, and are noticed and corrected immediately.

The most serious failure that has been encountered to date occurred when lightning struck near a pair of buildings that had an Ethernet cable strung underground between them. The input

transistors in a dozen or so transceivers became fused. Finding all the shorted transceivers took several hours. Partitioning the network made it apparent that there were problems in all sections. A time domain reflectometer (TDR) was helpful in locating the damaged transceivers among the 100 or so units connected to that network.

The likely cause of failure was a shift in ground potentials between the two buildings, which was coupled into the cable shield through several tap blocks that were found to be resting against grounded cable trays. The voltage drop along the length of the shield was not matched by a corresponding drop along the center conductor, and consequently half of the voltage present between the building grounds was also present between the center conductor and shield of the coax cable. This voltage damaged the input transistors in the transceivers. An improved design would include insulated tap blocks and connector housings as well as current-limiting resistors in the transceiver input circuits.

Occasionally a controller malfunctions in such a way as to transmit continuous, correctly phase-encoded data. (As explained earlier, the transceiver prevents the controller from simply jamming the channel "on".) Failures can also be caused by malfunctioning software; for example, a program that repeatedly transmits broadcast packets can have a devastating effect on the performance of every host on the network.

Uncontrolled pollution of this nature can be prevented by introduction of a watchdog timer that electrically disconnects the transceiver from the bus unless periodically reset by the software. Such a scheme has been used in the Queen Mary College ENET and CNET in a manner that also verifies the proper functioning of the controller and low-level software [West & Davison, 1978].

Isolated failures. The frequency of packets lost due to interface-detected errors (incorrect CRC or termination at other than a word boundary) is about one in 6000 when averaged over many machines. This is so small as to have negligible effect on overall performance. Furthermore, an overwhelming majority of these errors are attributable to defects in the Alto Ethernet interface rather than to corruption on the wire. A redesigned interface, which features digital rather than analog phase decoding, has reduced the frequency of damaged packets to less than one in 2 million [Shoch & Hupp, 1979].

The electrical signalling used on the cable is such that the sync bit of a packet barely passes the DC receiver threshold after traversing one kilometer of cable, due to the stretching of pulse rise-times [Wigington & Nahman, 1957]. The average DC level is achieved only after several bit times. This results in occasional failure of a receiver to acquire correct bit phase. A worthwhile modification to the signalling protocol would be to precede each packet's sync bit with a fairly long preamble. This would permit higher speed operation while still using a DC receiver threshold. It would also permit phase decoding by means of a phase locked loop, which requires more than one bit time to lock onto the signal.

Higher-level protocols are resilient enough to mask rather gross hardware failures. For example, in a few instances a host's carrier sense circuit has failed, causing its transmitter not to

defer to an ongoing transmission but rather to collide with it. These and similar failures have negligible effects on end-to-end communication—even in the machines that have failed—and are detected only by careful network monitoring and measurement.

In the same vein, a host with a marginal transceiver or controller is sometimes unable to communicate with a particular set of other hosts but has no difficulty communicating with the rest. This is due primarily to variations in analog components.

Detecting this type of failure requires attempting communication between all n^2 pairs of hosts in the network. This may be done by a distributed diagnostic program, called a "worm", which runs in several Altos at a time and determines whether they can communicate with each other. Each instance of the program periodically finds another idle Alto, bootstrap-loads that Alto with a copy of itself, and destroys itself. As the worm crawls around the network, it reports its findings to a server which maintains a record of the network's logical connectivity.

6. Conclusion

The Ethernet system design has been successful in the role intended for it: a high-performance local transport mechanism for an internetwork architecture. The simplicity of the basic design, combined with careful engineering of the critical components, has resulted in great flexibility and high reliability. We have attempted to highlight the important positive and negative features of the prototype implementation and to assess their impact on an operational local network.

Acknowledgments

The Ethernet system was originally designed and constructed by Bob Metcalfe and Dave Boggs. The traffic and error statistics summarized here were the result of extensive measurements performed by John Shoch and Jon Hupp. Numerous other people in Xerox PARC and SDD have contributed to the success of the Ethernet.

References

[Abramson, 1977]
N. Abramson, The Throughput of Packet Broadcasting Channels, *IEEE Transactions on Communications*, vol. Com-25 no. 1, January 1977.

[Agrawala *et al.* 1977]
A. K. Agrawala, R. M. Bryant, and J. Agre, Analysis of an Ethernet-like Protocol, *Proceedings of the Computer Networking Symposium*, IEEE(CS)/NBS, Gaithersburg, December 1977.

[Almes & Lazowska, 1979]
G. T. Almes and E. D. Lazowska, *The Behavior of Ethernet-like Computer Communications Networks*, Technical Report no. 79-05-01, Dept. of Computer Science, University of Washington, April 1979.

[Boggs *et al*, 1980]
David R. Boggs, John F. Shoch, Edward A. Taft, and Robert M. Metcalfe, Pup: An Internetwork Architecture, *IEEE Transactions on Communications*, vol. Com-28 no. 4, April 1980, to appear; also available as Xerox PARC technical report CSL-79-10.

[Cerf & Kahn, 1974]
Vinton Cerf and Robert Kahn, A Protocol for Packet Network Interconnection, *IEEE Transactions of Communication*, vol. Com-22 no. 5, May 1974.

[Du Pont, 1979]
Plenum Cable of TEFLON FEP, E. I. Du Pont De Nemours & Co., Inc., Plastic Products & Resins Department, Wilmington, 1979.

[McQuillan, 1979]
John M. McQuillan, Local Network Technology and the Lessons of History, *Local Area Communications Network Symposium*, Boston, May 1979.

[Metcalfe, 1973]
Robert M. Metcalfe, *Packet Communication*, Harvard Ph.D. thesis, M.I.T. Project MAC TR-114, December 1973.

[Metcalfe & Boggs, 1976]
Robert M. Metcalfe and David R. Boggs, Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM*, vol. 19 no. 7, July 1976.

[Needham & Schroeder, 1978]
Roger Needham and Michael Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM*, vol. 21 no. 12, December 1978.

[Saltzer & Pogran, 1979]
Jerome H. Saltzer and Kenneth T. Pogran, A Star-shaped Ring Network with High Maintainability, *Local Area Communications Network Symposium*, Boston, May 1979.

[Shoch, 1979]
John F. Shoch, *Design and Performance of Local Computer Networks*, Stanford Ph.D. thesis, University Microfilms, August 1979.

[Shoch & Hupp, 1979]
John F. Shoch and Jon A. Hupp, Performance of an Ethernet Local Network—a Preliminary Report, *Local Area Communications Network Symposium*, Boston, May 1979.

[Sunshine, 1977]
Carl Sunshine, Interconnection of Computer Networks, *Computer Networks*, vol. 1, 1977.

[Thacker *et al.*, 1979]
C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, Alto: A Personal Computer, *Computer Structures: Readings and Examples* (Siewiorek, Bell, and Newell, eds.), 1979, to appear; also available as Xerox PARC technical report CSL-79-11.

[Tobagi & Hunt, 1979]
F. A. Tobagi and V. B. Hunt, Performance Analysis of Carrier Sense Multiple Access with Collision Detection, *Local Area Communications Network Symposium*, Boston, May 1979.

[West & Davison, 1978]
Anthony West and Allan Davison, *CNET—a Cheap Network for Distributed Computing*, TR-120, Computer System Laboratory, Queen Mary College, March 1978.

[Wigington & Nahman, 1957]
R. L. Wigington and N. S. Nahman, Transient Analysis of Coaxial Cables Considering Skin Effect, *Proceedings of the IRE*, vol. 45, February 1957.

Measured Performance of an Ethernet Local Network

by John F Shoch and Jon A. Hupp

February 1980, reprinted October 1982

© Copyright Association for Computing Machinery 1980. Printed with permission.

Abstract: The Ethernet communications network is a broadcast, multi-access system for local computer networking, using the techniques of *carrier sense* and *collision detection*. Recently we have measured the actual performance and error characteristics of an existing Ethernet installation which provides communications services to over 120 directly connected hosts.

This paper is a report on some of those measurements—characterizing "typical" traffic characteristics in this environment—and demonstrating that the system works very well. About 300 million bytes traverse the network daily; under normal load, latency and error rates are extremely low and there are very few collisions. Under extremely heavy load—artificially generated—the system shows stable behavior, and channel utilization remains above 97%, as predicted.

A version of this paper appeared in *Communications of the ACM*, vol. 23 no. 12, December 1980.

CR categories: 3.81.

Key words and phrases: Local computer networks, Ethernet system, carrier sense multiple access with collision detection (CSMA/CD), packet switching, measurements.

XEROX

Xerox Corporation
Palo Alto Research Centers
3333 Coyote Hill Road
Palo Alto, California 94304

1. Introduction

One of the most attractive architectures for a local computer network is a shared, multi-access bus with distributed control. This approach was originally articulated in the design of the Ethernet communications network, in which all of the host machines share access to a single, passive coaxial cable [Metcalfe & Boggs, 1976].

A number of interconnected Ethernet system installations have been in operation for several years, providing communications service for many hosts and supporting a wide variety of applications. The networks have always functioned very well, and probably represent the largest collection of local networks in use today. The intent of the present study was to further our understanding of this system by characterizing the actual behavior of one of these installations in normal use: traffic volume, packet sizes, error rates, efficiency, and more. In addition, we have constructed a test environment that allows us to artificially generate extreme amounts of traffic -- well beyond the expected operating conditions -- in order to help assess performance under heavily loaded conditions.

Successful use of a local computer network depends on much more than the specific communication medium: applications are using higher level protocols, including a complete architecture for internetwork communication among many different systems [Boggs, *et al.*, 1980]. Those topics, however, are beyond the range of the current study, which is directed primarily at the behavior of the Ethernet system itself. A preliminary version of these results was reported in [Shoch & Hupp, 1979]; more detailed discussion will be found in [Shoch, 1980].

In the sections which follow, we review the Ethernet local network, describe the experimental environment and test methodology, measure the system under normal operating conditions, and then measure the behavior with artificially high loads.

2. Review of the Ethernet principles

The approach used in the Ethernet system traces its roots back to the radio-based Aloha packet switching network, developed at the University of Hawaii [Abramson, 1970]. In the Aloha network, terminals equipped with packet radios shared a common multi-access radio channel. There was, however, no central controller allocating access to the channel; instead, a *random access* procedure was used, in which each terminal independently decided when to transmit. With this simplest *pure Aloha* scheme, two terminals may transmit at the same time producing a collision; after a collision, each terminal will have to wait for a random interval and then retransmit. The attraction of this approach is the elimination of any central control; as the load increases, however, the maximum possible utilization of a pure Aloha channel is only 18% [Abramson, 1977].

Unlike the pure Aloha scheme, when an Ethernet station wishes to send a packet a *carrier sense* mechanism is first used, forcing the station to *defer* if any transmission is in progress. If no other station is transmitting, the sender can begin immediately, with zero latency; otherwise, the sender

waits until the packet has passed before transmitting.

It is possible that two or more stations will sense that the channel is idle and begin transmissions simultaneously, producing a *collision.* Each sender, however, continues to monitor the cable during transmission, and can provide *collision detection* when the signal on the cable does not match its own output. In that case, each station stops transmitting, uses a *collision consensus enforcement* procedure to ensure that all other colliding stations have seen the collision, and then stops. A retransmission is then scheduled for some later time.

To avoid repeated collisions, each station waits for a random period of time before retransmitting. To avoid overloading the channel, thus making it unstable, the range of this retransmission interval is increased under times of heavy load, using a *binary exponential backoff* algorithm.

Taken together, these mechanisms represent the Ethernet random access procedure, also referred to as *carrier sense multiple access with collision detection,* or CSMA/CD. This access procedure can be applied to any suitable broadcast channel, such as radio, twisted pair, coaxial cable, fiber optics, diffuse infrared, or others. Coaxial cable is well suited for use in constructing a local computer network -- a system designed to provide high-bandwidth communications for machines within a distance of approximately 1 kilometer.

The shared component of an Ethernet local network consists of a single coaxial cable, typically strung in a meandering fashion through a building, perhaps in the ceiling or under a raised floor. As shown in Figure 1, individual computers, or *stations,* connect to the cable with the use of a CATV-style *tap;* a small *transceiver* is connected at the tap, with a cable running down to the *interface* which might be located in the station. The use of a passive medium, and the lack of any active elements in the shared portion, combine to help provide a very reliable and flexible system. This approach also provides for easy reconfiguration of stations: machines can be moved -- disconnected from one point and reattached at another -- without any need to take down the network. (For further details on the design of the Ethernet system, see [Metcalfe & Boggs, 1976; Metcalfe, *et al.,* 1977; Shoch, 1979, 1980; and Crane & Taft, 1980].)

All of these mechanisms are combined to increase the probability that an Ethernet system will correctly deliver a client's packet. The network gives its best effort, but -- like any other low-level communications subsystem -- the Ethernet communications network cannot provide a 100% guarantee of delivery. In a complete network architecture, suitable *packet protocols* will be layered on top of this local network, to provide further reliability.

3. Experimental environment

The principles of the Ethernet system were first realized in the *experimental* or *prototype* system developed at the Xerox Palo Alto Research Center: using regular coaxial cable, running at 2.94 Mbps. There are now many of these networks in use in locations throughout the United States, tied

together with the Pup internetwork architecture [Boggs, *et al.*, 1980].

The particular local Ethernet network installation chosen for these measurements is one of the oldest in existence: it spans about 1800 feet (550 m.) and connects over 120 machines. These machines include large numbers of single-user stand-alone computers (mainly the Alto computer [Thacker, *et al.*, 1980]), two time-sharing servers running the Tenex operating system, numerous shared printers and file servers, as well as several internetwork gateways.

Applications which use the network include file transmission to the printers, file transfer to one of the storage systems, specialized multi-machine programs, access to shared data bases, remote diagnostics, down-line loading of programs, terminal access to the time-sharing machines, and others.

To conduct the measurements, a series of specialized test and monitoring programs has been built to assess the behavior of the network. The broadcast nature of an Ethernet system makes it particularly well suited for the passive collection of measurements: an individual station can sense the state of the cable and -- by operating its interface in a *promiscuous* mode -- can receive all of the packets passing by. (We thus avoid many of the Heisenberg problems: no processing is done at the source or destination to collect statistics, and no communications bandwidth is used to report them.)

Most of the measurements in the next two sections were collected using this passive technique, watching packets that go by on the cable.

4. Reliability and packet error rates

One of the major objectives of any local network is to provide reliable communications facilities, reflected both in the continued availability of the network itself, and in the lowest possible error rate as seen by individual hosts. Overall reliability was one of the most important considerations in the development of the Ethernet network. Thus, the only shared component in the communications subsystem is the passive coaxial cable; the shared portion has no electronics, no active components, no switches, no power being supplied through the line, and no power supplies.

In practical operation, there have been very few system-wide failures; almost all of these are caused by human error of some sort -- for example, removing the terminator at the end of the cable. On one occasion, a lightning strike generated a large surge in the building ground, disabling a number of improperly installed transceivers. Modified installation procedures and a minor circuit change have now significantly improved lightning resistance.

These system-wide failures are easy to control, and overall reliability is very high. Individual packets, however, are subject to damage when being transmitted, and may arrive in error. The design of the Ethernet system does not attempt to guarantee 100% error-free operation, and applications may use suitable error control procedures, such as packet acknowledgments and

retransmissions. But, for sustained high performance, one would certainly want to minimize the occurrence of packet errors.

There are several mechanisms available to detect and discard a damaged packet. When a station receives a packet, the interface checks that an integral number of 16-bit words has been received and that the 16-bit cyclic redundancy checksum (CRC) is correct. If either of these conditions is not met, the packet is considered damaged and is normally discarded.

With a passive listener set to receive every packet on the net, our current machines initially reported one damaged packet in about 6000, but there was wide variance among machines. Further examination has indicated that most of these errors reflect problems in the receiver section of the interface itself, and the packets were actually well-formed on the cable. Experimentation with a revised interface has produced normalized packet error rates of about 1 in 2,000,000 packets. Since these error rates are so low, packets received in error have been excluded from all of the results reported below.

We should note that there is a much more significant cause of lost packets which has nothing to do with network performance: failure to keep a receiving interface running properly. Some networking systems have proposed use of an interface which will receive a packet and then have a lengthy *refractory period* or *blind spot* before the receiver is restarted; these devices have a long *receiver-to-receiver turnaround time.* During these intervals, any packets sent to this station may be correct on the cable, but entirely missed by the station. Interface turnaround times require careful attention in the design of network equipment (see [Shoch, 1980]); but this is a problem beyond the scope of the local network itself.

5. Performance characteristics under normal load conditions

5.1. Overall traffic characteristics

In normal use, this single Ethernet system carries about 2.2 million packets in a 24-hour period, totalling almost 300 million bytes. (For comparison, this traffic roughly corresponds to about half of the volume carried through the Arpanet on an average day.) Not surprisingly, the load is very light at night and is heaviest during the regular daytime hours, with a slight dip around lunchtime; see Figure 2, showing network load over a 24-hour period, sampled over fairly long intervals (six minutes each).

5.2. Utilization

Again measured over a full 24-hour day, this traffic represents an extremely modest usage of the net: ranging from about 0.60% to as much as 0.84%. During shorter periods, however, maximum utilization in the busiest interval is much higher: about 3.6% over the busiest hour, 17%

over the busiest minute, and 37% in the busiest second. Figure 3 represents the load observed over four minutes on the Ethernet network, with individual samples summed over one second intervals; note that the full range of this figure would be contained within just one sample in the previous figure.

These results help to verify our design assumption that computer communications applications tend to produce a "bursty" pattern of requests; what we observe on the shared Ethernet channel is the aggregation of some number of independent sources of "bursty" traffic.

5.3. Packet length

Packets sent through the system exhibit a bimodal distribution of packet length: most of the packets are short ones (containing terminal traffic, acknowledgments, etc.), but most of the total volume is carried in the large packets (often containing file transfer traffic) -- see Figure 4. This is similar to some of the measurements reported for the Arpanet [Kleinrock & Naylor, 1974], but very different from the distributions frequently used in analytical models of networks.

To some extent, the upper and lower limits on the length of packets represent artifacts of the various implementations. Almost all of the traffic consists of Ethernet packets carrying encapsulated internetwork packets, or Pups [Boggs, *et al.*, 1980]. Thus, the minimum sized packet with no data would usually include the Ethernet and Pup headers, while acknowledgments and packets with only 1 or 2 bytes of data would be just a bit larger. At the other extreme, software considerations usually impose an effective upper limit on the size of a packet; depending on the particular system, this ranges from about 200 to 540 bytes.

The mean packet length is about 122 bytes, and the median is about 32 bytes.

5.4. Source-destination traffic patterns

On a given day, over 120 hosts use this Ethernet installation -- nearly every machine known to be connected. The extent of this utilization ranges from a fraction of a percent to over 25% of the observed network traffic. Nearly 1300 different source-destination pairs communicate during this period; on the average, each host sends packets to more than 10 other hosts, but some of this traffic is concentrated to and from specialized servers (see the following section).

Figure 5 shows the source-destination traffic matrix for a fairly typical day. A heavily used server both sends and receives packets from many other hosts; each server appears in the figure as a pair of broken lines, intersecting on the diagonal and roughly symmetric about that point.

This matrix only indicates sources and destinations on the local Ethernet system, but some of these servers are gateways to other networks. Thus, they represent the path to some larger number of internetwork addresses, and there are many more internetwork source-destination pairs.

5.5. Server traffic patterns

As suggested from the source-destination traffic matrix, many of the packets are going to or from specialized servers: time-sharing systems, gateways, information servers, file systems, and printers.

Over one typical day, these identifiable servers sent about 69% of the packets, and received about 73%.

5.6. Inter-packet arrival times

The distribution of inter-packet arrival times over a 24-hour period is shown in Figure 6. The mean inter-packet arrival time is 39.5 ms., with a standard deviation of 55.0 ms. and a median of 8.5 ms.

A fair portion of the current traffic consists of request/response transactions with a server. With low utilization, it is not uncommon to have this exchange take place with no intervening packets being transmitted; some of the spikes in the inter-packet arrival time distribution represent the turnaround times of these servers.

Figure 7 indicates the cumulative distribution of inter-packet arrival times: it shows that 50% of the packets are followed by the next packet within 10 ms., 90% of the packets are followed by the next packet within 64 ms., and 99% within 183 ms.

5.7. Latency and collisions detected by a sender

With the current level of traffic, it is not surprising that most attempts to send a packet succeed on the first transmission -- there are no collisions and no need to backoff and retransmit.

To measure this result, a test program has been run which periodically wakes up and tries to transmit a packet; under normal load, it indicates that 99.18% of the packets make it out with zero latency, 0.79% of the packets are delayed due to deference, and less than 0.03% of the packets are involved in collisions.

5.8. Overhead

In addition to useful data, there are several forms of overhead which impact Ethernet communications: headers and error checking on every packet, as well as entire packets carrying acknowledgments, routing tables, or other ancillary information.

The overhead due specifically to Ethernet encapsulation is only 4 bytes; this represents about 3% of an average length packet. The standard internetwork protocol header consumes another 22 bytes, or about 18% of an average packet. Thus, data fields of all packets represent about 79% of the bits carried. It is worth noting that overhead accounts for less than 5% of the maximum length packet used for high-volume data transfers.

To get a precise measure of the actual data bits sent by user protocols one would need to exclude acknowledgment packets, and perhaps model both ends of the end-to-end process. As an approximation, however, we can obtain a reasonable estimate by measuring the amount of data carried inside data packets of the most common protocols. This will generally provide a conservative estimate of user traffic, counting as overhead all of the packets used to set up a connection, error packets, boot loader packets, and all of the specialized packet types exchanged by other user protocols. (This procedure may also double-count the data in any packets which are actually retransmitted by the source; in general, that number is very small.)

Using this alternative approach, about 69% of all the bits carried through the Ethernet system have been classified as user data, leaving a total of about 31% encompassing all forms of overhead.

5.9. Intranet vs. internet traffic

About 72% of all of the packets seen have both their source and destination on this network -- they are intranet packets. About 28% of the packets are internet traffic, coming from or going to another network via an internetwork gateway. The presence of the two large time-sharing systems on this particular network accounts for much of this traffic, as users access these hosts from distant locations. In addition, it is possible for a local Ethernet system to serve as a transit network for traffic originating on one network and destined for another; with the current topology, however, we see few such "transit packets" on this particular network.

6. Performance characteristics under high load conditions

The foregoing discussion has considered network performance under normal operating conditions, reflecting the current demands placed upon the network by the existing set of applications. Further growth of new systems will increase the load on the net, and the system should certainly be able to accommodate short term bursts of very high load. One of the considerations in designing an application would be the performance of an Ethernet system as the load increases dramatically.

Initial modelling of the Ethernet system approach [Metcalfe & Boggs, 1976] has indicated that the system should continue to function very well as the load goes up. The following measurements serve to verify that prediction.

In a high load situation, each individual host can be set to generate packets at a specified rate, generally less than the channel capacity, and independent of the rate at which these packets are actually sent through the channel. The total offered load then represents the sum of these individual offered loads, and may be less than 100% of the channel capacity, or more than 100% (in which case some of the transmit queues overflow, and packets are lost). This load can be varied by changing the number of hosts, or the load generated per host. (In section 7 below, we explore a bit

further the special case in which each host can be viewed as a continuously queued source, always trying to transmit a packet, and generating a new one as soon as this one is sent. Thus, each host can be viewed as attempting to present a load of 100%. In this case, it is only the number of hosts attempting to transmit which is of interest; it is not meaningful to talk about increasing the load generated by each host.)

Measurements of interest include the actual level of channel utilization, the stability of the system as the load increases (does the total network utilization dramatically decrease as the load increases?), and the fairness with which the channel is shared.

To help gauge this behavior under high load, and to help stress the capabilities of the system, we have constructed test programs which generate artificially high levels of traffic. Using a special control program, we use the network to find idle machines and then load them with a test program which can be set to produce a specified offered load to the network. Statistics are accumulated as to the number of packets successfully sent and at the end of a test period, after all the test machines are idle, the controlling machine can reach out and collect these statistics.

When a test program generates a packet for transmission, it is placed upon an output queue; actual transmission is performed by an output process, which also handles retransmissions due to collisions. The presence of the queue allows the sender to tolerate short term variations in the availability of the channel. If the total offered load exceeds 100%, however, it is clear that some of the traffic being generated will never be transmitted successfully, and it merely gets dumped off of the output queue.

One can start with a modest offered load, and then increase that level by either adding more machines or increasing the load being offered by each host. With ideal scheduling, one would like to see the total channel utilization increasing with the total offered load, up to 100% (see Figure 8). Beyond 100% load -- under *very* heavy load -- we would like to see the channel utilization remain at 100%, representing full use of the available capacity.

Few real system can perform this well. A pure Aloha channel gets 18% utilization at best and a slotted Aloha channel can realize 37% utilization. Both Aloha mechanisms can become unstable: utilization declines significantly as offered load increases [Abramson, 1977]. In general, one tries to use a control procedure that will maximize the utilization, while remaining stable as the load increases.

Actual test runs were made on the regular Ethernet system described above, usually at night, when there was very little other traffic. The high load results reported below were for full-size data packets (512 bytes), typical of most of the volume moved through the net.

6.1. Maximum utilization

As the total offered load increases from 0% to 90%, channel utilization matches it perfectly: all of the traffic gets out correctly. As the offered load moves above 90%, the channel utilization flattens out at a level above 96% (see Figure 9). This experiment has been run with even greater

loads, as high as 9000% (well beyond any reasonable operating region!); utilization continued to increase slightly, approaching 98% -- just 2% short of the ideal case.

6.2. Stability

As Figure 9 indicates, an Ethernet system under high load shows no instability: the throughput curve does not decline as the total offered load increases.

6.3. Fairness

The Ethernet control discipline is also very fair in its allocation of channel capacity: at 100% offered load (10 hosts at 10%/host) we observe a total utilization of 94% with individual throughputs ranging from 9.3 to 9.6%.

7. Performance characteristics with continuously queued sources

The preceding section described a general model in which both the load per host and the number of hosts could be varied, producing total offered load in the range of 0% to 150%. In this section we examine a bit more closely the special case in which every host is continually queued, trying to generate a load of 100% of the channel rate.

This corresponds to the analytical model developed in the original paper on the Ethernet communications network (see [Metcalfe & Boggs, 1976]); the actual measurement results are of use in trying to verify this model. In the ideal case, a single machine continuously transmitting would produce a utilization of 100%; with perfect scheduling, the addition of other machines would not cause the total utilization to decrease. But we do not have perfect knowledge nor scheduling, and as more users queue up to transmit, collisions take place, and the utilization decreases. The analytical results have predicted, however, that for reasonably large packet sizes the distributed control algorithm of the Ethernet would only cause the total utilization to decrease by a small percentage (see Figure 10a).

In practice, we can only approximate these test conditions, since the channel cannot be driven continuously by one host (due to non-zero transmitter-to-transmitter turnaround times). A single machine transmitting large 512 byte packets, for example, can offer a 95% load, but with smaller packet sizes the effective load becomes even lower. With the current population of machines, the most interesting (and feasible) measurements use from 5 to 64 hosts.

7.1. Maximum utilization

As predicted, the total utilization starts out very high, and decreases somewhat as additional hosts are added. The test results for various combinations of packet length and number of hosts are summarized in the table below (also see Figures 10a and 10b):

Ethernet utilization with continuously queued sources
Q = number of hosts, each generating 100% load

	Packet length P (in bytes)				
Q	512	128	64	6	4
5	97%	95%	94%	72%	----
10	97%	91%	89%	68%	58%
32	97%	90%	83%	64%	56%
64	97%	92%	85%	61%	54%

These measured results are very similar to the results predicted in the model used in the original paper on the Ethernet system. As expected, Ethernet utilization increases with packet size. For full-size data packets -- 80% of the typical traffic -- total utilization remains above 97%, even with 64 continuously queued hosts.

With small packets, any time lost to collisions and collision resolution becomes larger compared to the packet size, and total utilization decreases. Four bytes represents the smallest packet we can send through the Ethernet network, and utilization would decrease further if the packets were even smaller. Indeed, with even smaller packets the packet transmission time should approach the *collision interval* and network utilization approaches 1/e, the maximum efficiency of a slotted Aloha network [Abramson, 1977].

7.2. Stability

As the table above shows, the Ethernet system remains stable even under extreme overload conditions. In experiments with as many as 90 hosts sending medium- to large-size packets (each offering up to a 95% network load) Ethernet utilization remains high, and shows no signs of suddenly decreasing or becoming unstable.

7.3. Fairness

With more than one continuously queued source, some of the traffic cannot be accommodated, and each station can get only some fraction of its nominally desired bandwidth (100% per host). In the full range of tests the system exhibits very good fairness to all of the machines. With 90 hosts sending, the average utilization per host is 1.1%, ranging from about .9% to 1.3%.

8. Concluding remarks

From these results, we can now begin to verify some important hypotheses about the performance of an Ethernet local network:

1. The error rates are very low, and very few packets are lost.
2. Under normal load, transmitting stations rarely have to defer and there are very few collisions. Those few collisions are resolved very quickly. Thus, the access time for any station attempting to transmit is virtually zero.
3. Under heavy load there are more collisions, but the collision detection and resolution mechanisms work well, and channel utilization remains very high -- almost 100%. In addition, channel sharing is quite fair.
4. Even under extreme overload, the Ethernet channel does not become unstable. (There have been several proposals for complex control schemes which have claimed higher utilization than the Ethernet approach, fairer allocation of the channel, or better performance [Shoch, 1979]. In light of the above results, these alternatives offer increased complexity with little potential benefit.)

With all of these characteristics, an Ethernet system remains a particularly attractive choice as an architecture for local communication.

9. Acknowledgments

The original design of the Ethernet system is due primarily to Robert M. Metcalfe and David R. Boggs; they have both been of great help in undertaking this project. We are also indebted to Edward A. Taft for his help in conducting this work.

10. References

[Abramson, 1970]
N. Abramson, "The Aloha system -- another alternative for computer communications," *AFIPS Conference Proceedings (FJCC 1977)*, 37, 1970, pp. 281-285.

[Abramson, 1977]
N. Abramson, "The throughput of packet broadcasting channels," *IEEE Transactions on Communications*, com-25:1, January 1977, pp. 117-128.

[Boggs, *et al.*, 1980]
D. R. Boggs, J. F. Shoch, E. A. Taft, and R. M. Metcalfe, "PUP: An internetwork architecture," *IEEE Transactions on Communications*, com-28:4, April 1980.

[Crane & Taft, 1980]
R. C. Crane and E. A. Taft, "Practical considerations in Ethernet local network design," *Proc. of the 13th Hawaii International Conference on Systems Sciences*, January 1980, pp. 166-174.

[Kleinrock & Naylor, 1974]
L. Kleinrock and W. E. Naylor, "On measured behavior of the ARPA Network," *AFIPS Conference Proceedings (NCC 1974)*, 43, June 1974, pp. 767-780.

[Metcalfe & Boggs, 1976]
R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *CACM*, 19:7, July 1976, pp. 395-404.

[Metcalfe, *et al.*, 1977]
R. M. Metcalfe, D. R. Boggs, C. P. Thacker, and B. W. Lampson, *Multipoint data communication system with collision detection*, United States Patent No. 4,063,220, December 13, 1977.

[Shoch, 1979]
J. F. Shoch, *An annotated bibliography on local computer networks (1st edition)*, Xerox Parc Technical Report SSL-79-5, and IFIP Working Group 6.4 Working Paper 79-1, October 1979.

[Shoch, 1980]
J. F. Shoch, *Design and performance of local computer networks*, in press, to appear 1980.

[Shoch & Hupp, 1979]
J. F. Shoch and J. A. Hupp, "Performance of an Ethernet local network -- a preliminary report," *Local Area Communications Network Symposium*, Boston, May 1979, pp. 113-125. Revised version presented at the *20th IEEE Computer Society International Conference (Compcon '80 Spring)*, San Francisco, February 1980.

[Thacker, *et al.*, 1979]
C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, *Alto: A personal computer*, Xerox Parc Technical Report CSL-79-11, August 1979. To appear in Siewiorek, Bell, and Newell (Eds.), *Computer Structures: Readings and Examples, 2nd edition.*

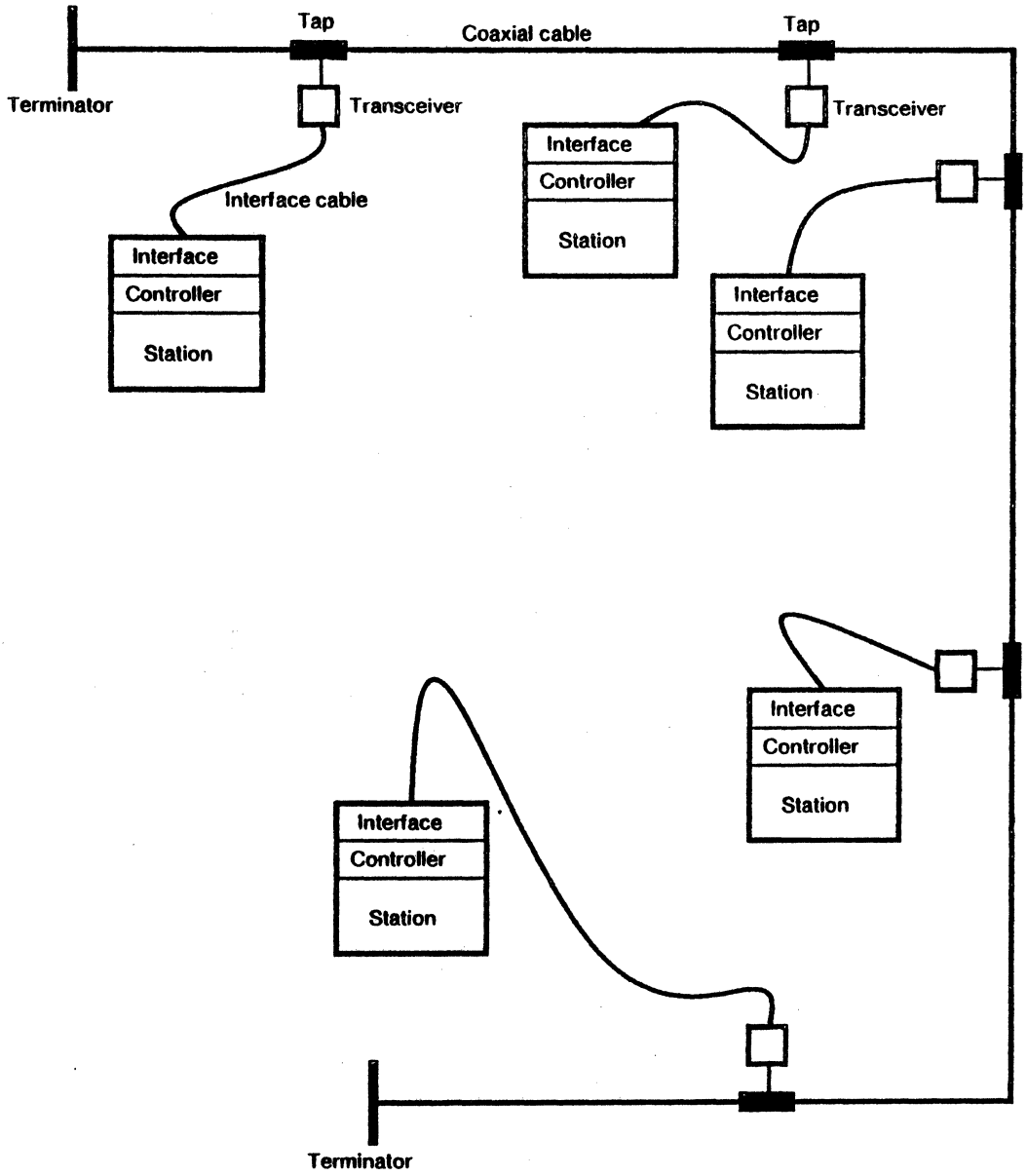
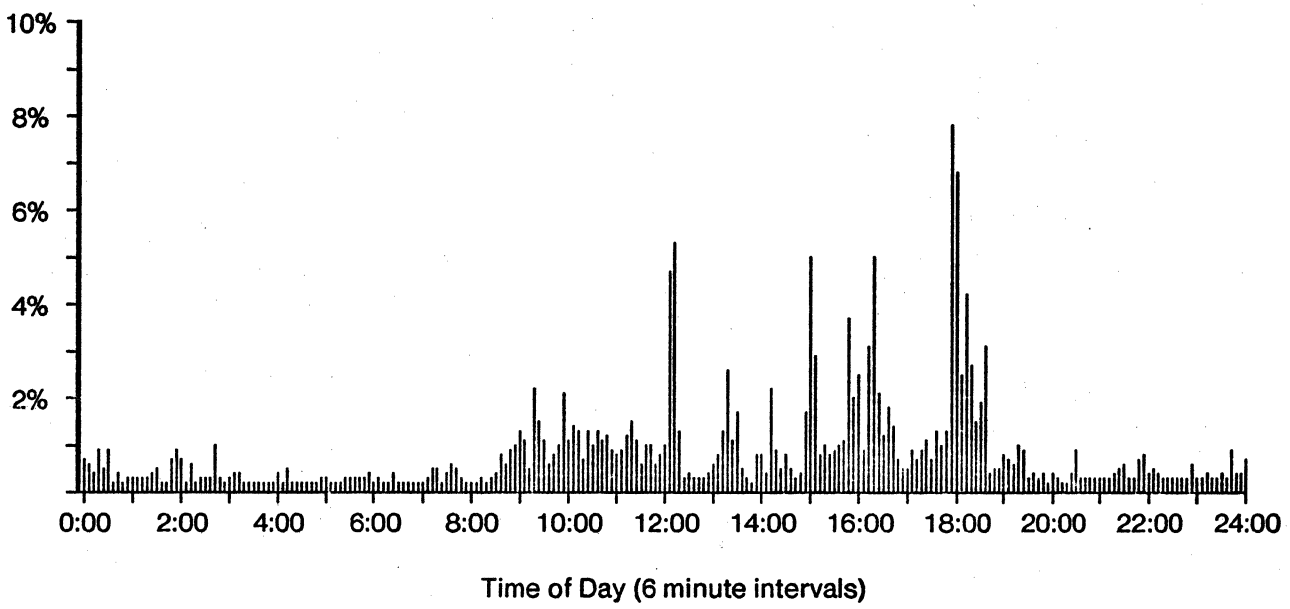


Figure 1: An overview of an Ethernet communications network (after [Metcalfe & Boggs, 1976]).



Max Load This Period = 7.9%
Min Load This Period = 0.2%
Average Load This Period = 0.8%

Figure 2: Ethernet load on a typical day, sampled over 6 minute intervals.

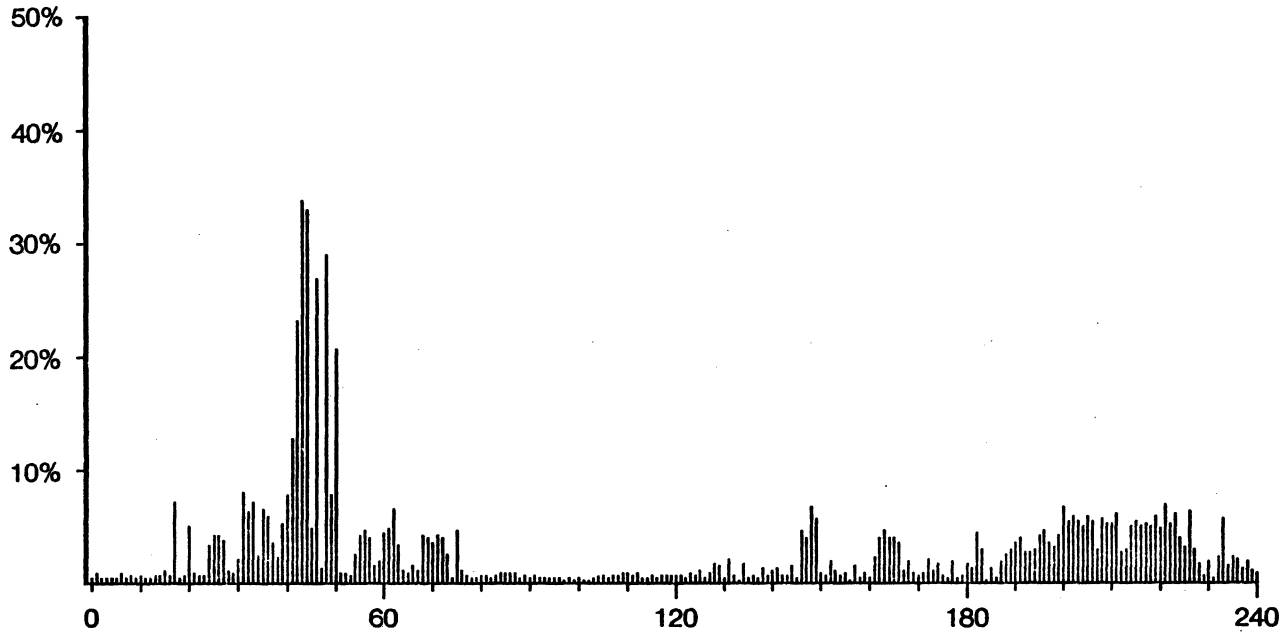


Figure 3: Ethernet load over a 4 minute period, sampled over 1 second intervals.

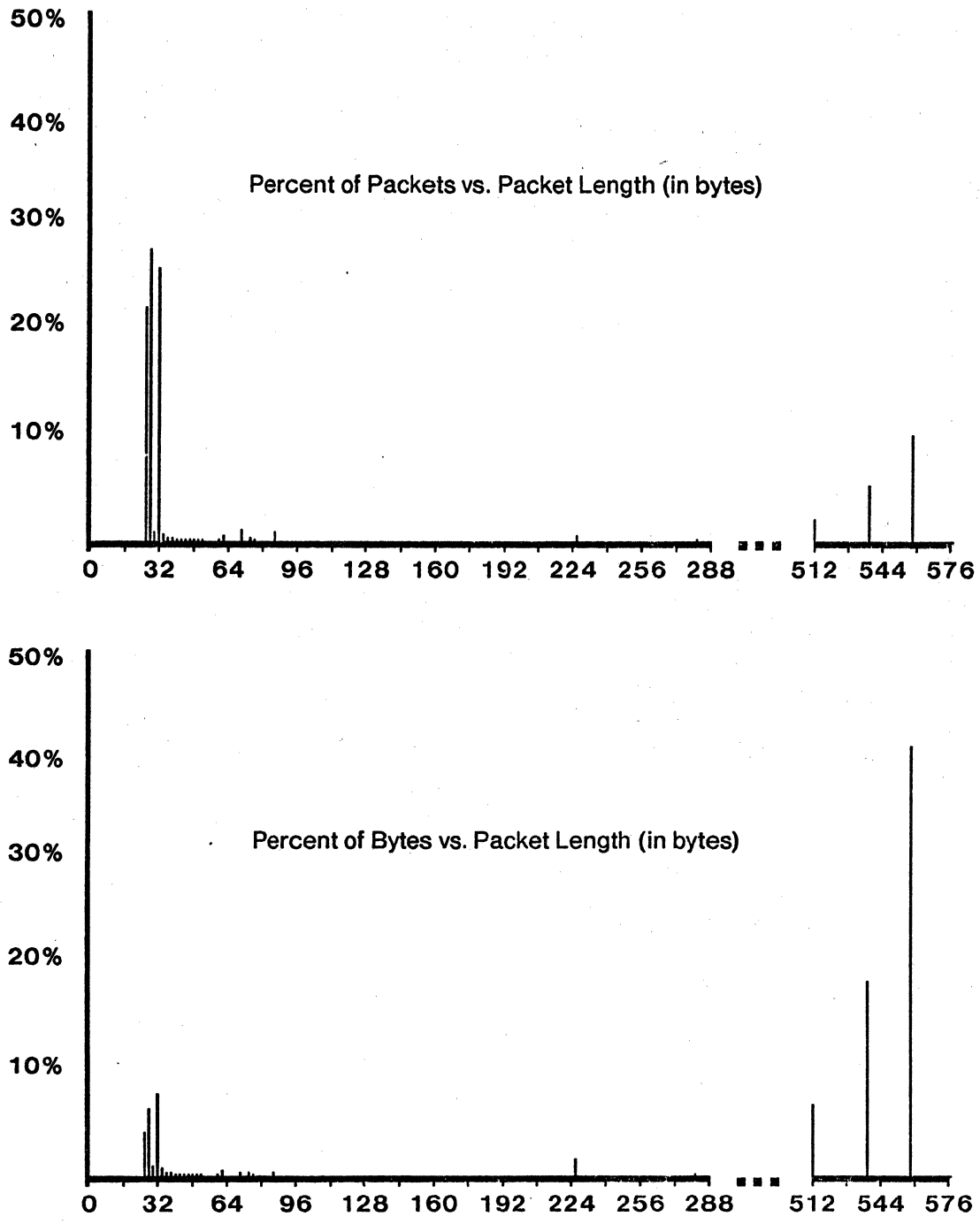


Figure 4: Distribution of packet lengths.

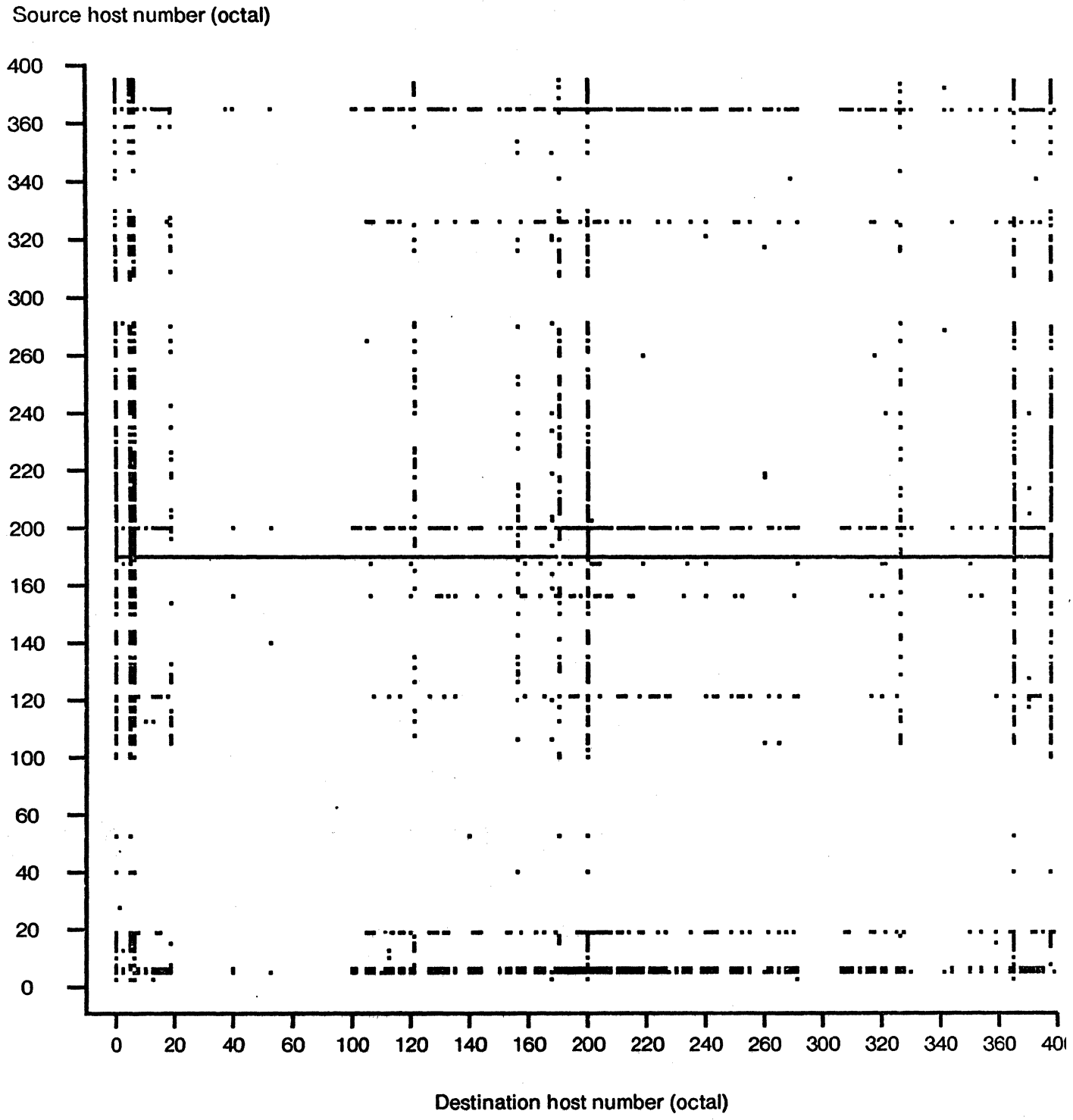


Figure 5: Source-destination traffic matrix.

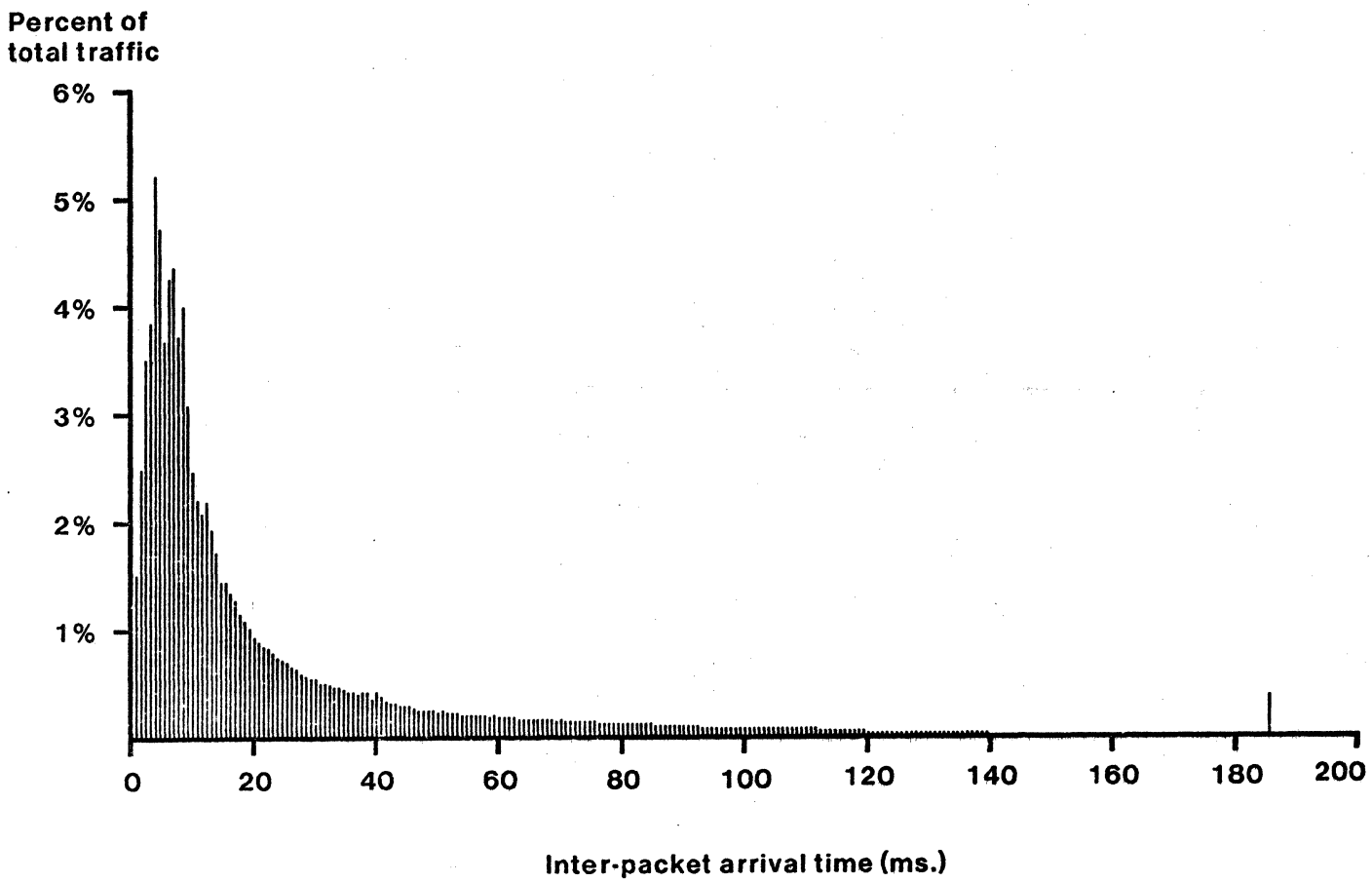


Figure 6: Distribution of inter-packet arrival times.

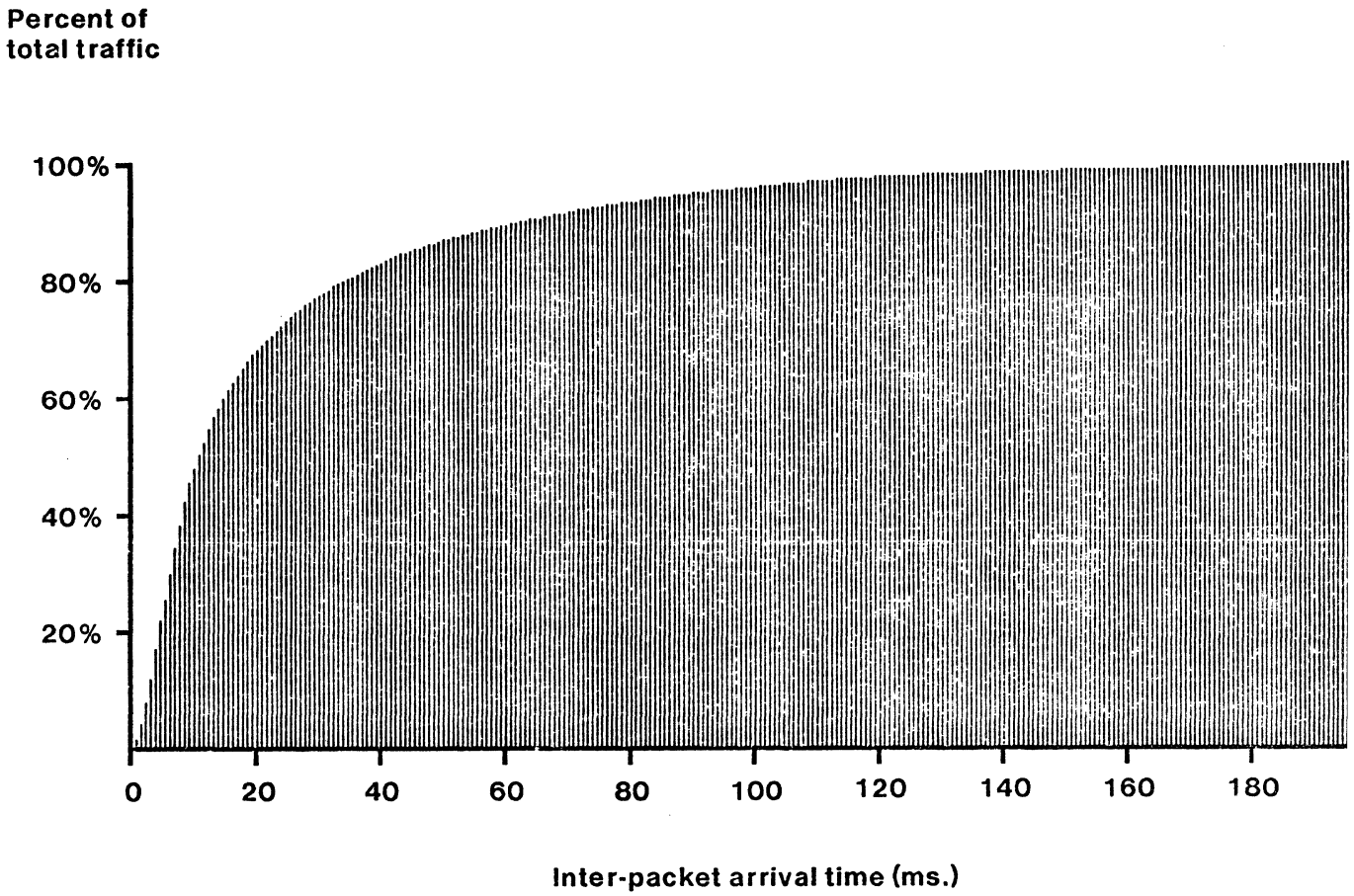


Figure 7: Cumulative inter-packet arrival times.

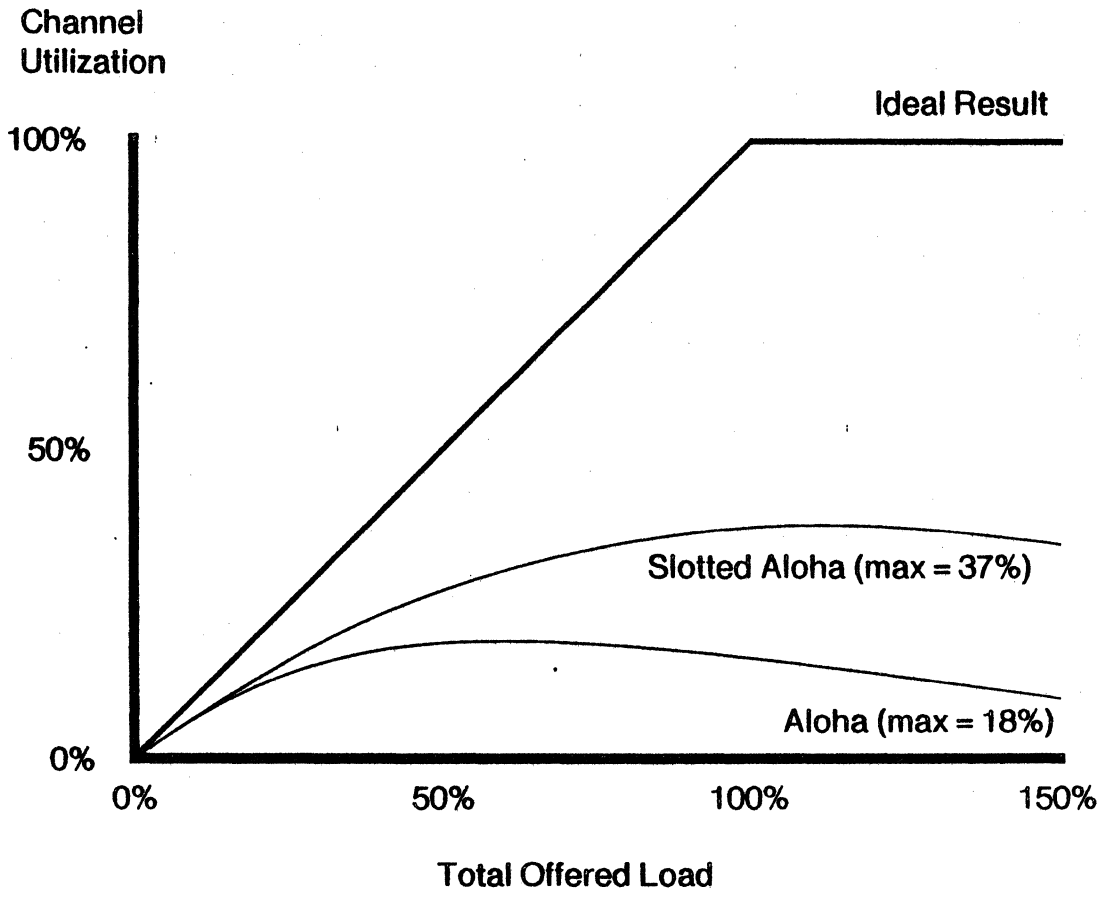


Figure 8: Utilization for several Aloha schemes.

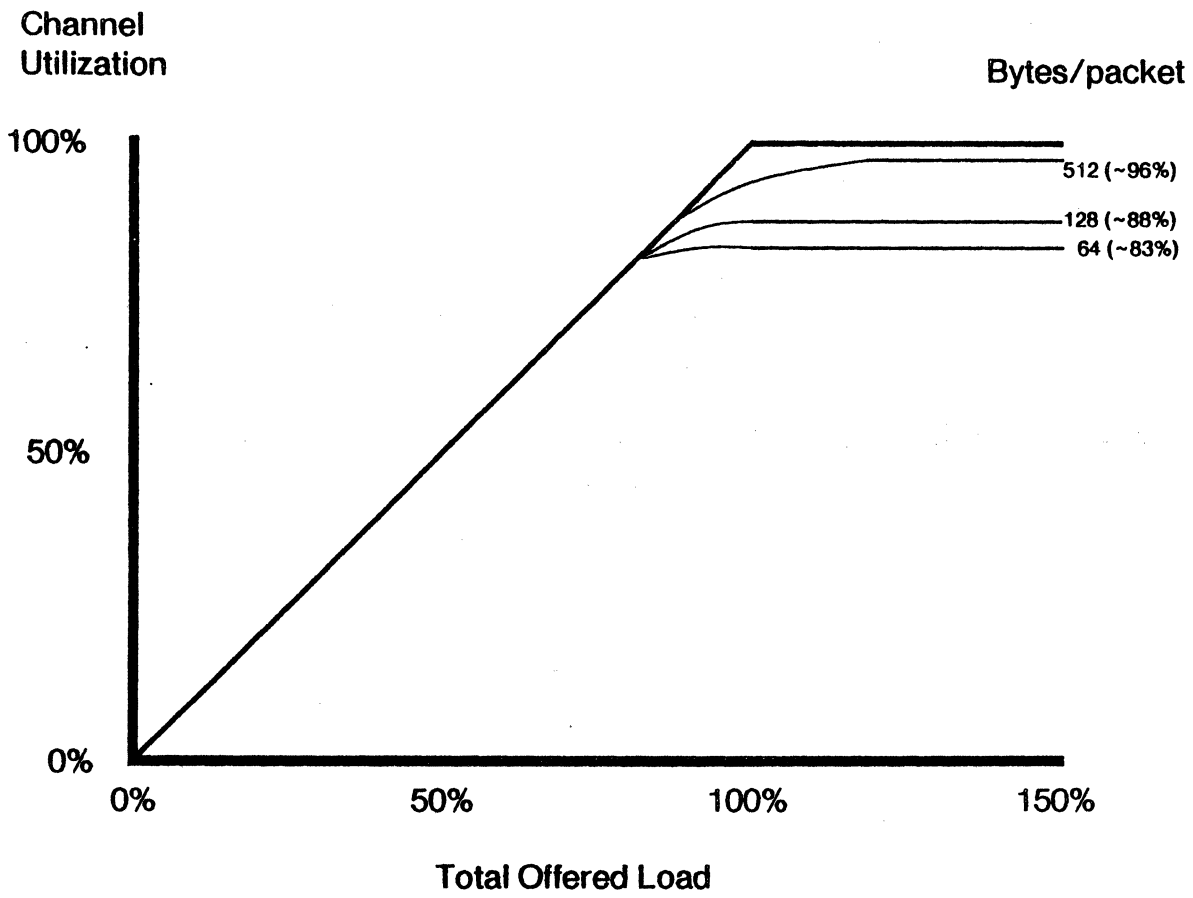


Figure 9: Measured utilization of the Ethernet network under high load.

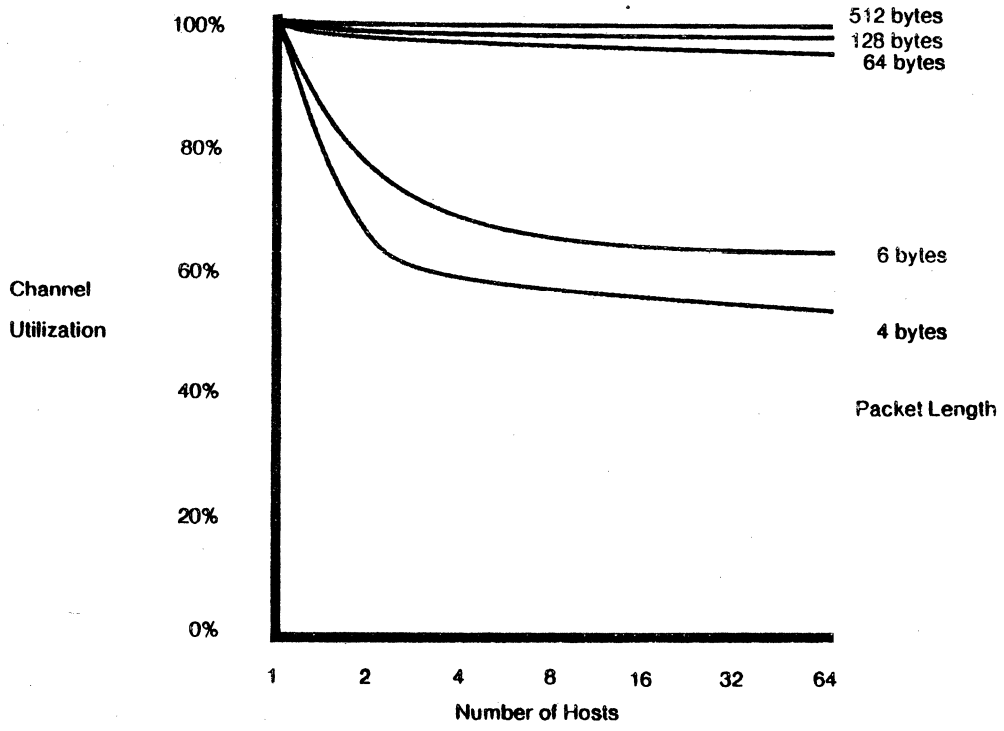


Figure 10a: Predicted utilization with continuously queued sources (recomputed from [Metcalfe & Boggs, 1976]).

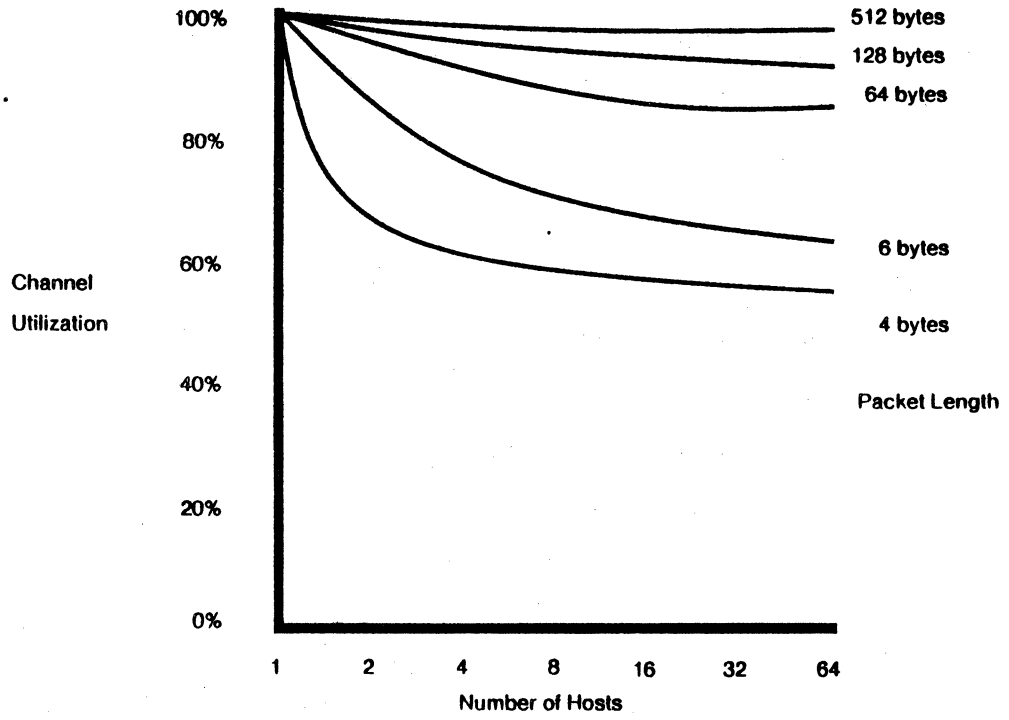


Figure 10b: Measured utilization with continuously queued sources.